TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SANTIAGO ROS NAVARRO

HARDWARE AND SOFTWARE DESIGN OF A WIRELESS ELECTRONIC CONTROLLER FOR DIGITAL SLOT CAR APPLICATIONS

Master of Science thesis

Examiner: prof. Jari Nurmi
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 8th April 2015

ABSTRACT

SANTIAGO ROS NAVARRO: Hardware and Software Design of a Wireless Electronic Controller for Digital Slot Car Applications
Tampere University of Technology
Master of Science Thesis, 127 pages, 19 Appendix pages
May 2015
Master´s Degree Programme in Electrical Engineering
Major: Industrial and Electronic Engineering
Examiner: Professor Jari Nurmi

Keywords: digital slot car, slot car racing, controller, software, hardware.

The main objectives of this Master of Science Thesis project are the design, development and implementation of an innovative wireless electronic controller for digital slot car applications. A slot vehicle is an electrical powered model that is guided by a groove or slot in the track on which it runs. The slot controller designed allows an accurate control of the change lane, brake and throttle actions in slot vehicles and it resolves the present problems of commercial slot car controllers.

This project focuses on the complete development of the electronic hardware interface dedicated to control digital slot vehicles. The development was performed using an electronic platform that combines a low cost and high performance microcontroller, which is equipped with conditioning electronics necessary to achieve the adjustment of slot vehicles parameters. The control interface, conditioning electronics and the necessary electronics for wireless communication are topics to considerer.

The development of the system software is also broached in this project, which is responsible for providing different driving modes possibilities for users depending on the performance required by the characteristics of each slot racing circuit. For this development, the main slot car controllers on the market have been analysed and the most common components in the multifunction control units of the circuits to perform a design that implements a microcontroller, which gives intelligence and enables to have an advanced control in digital slot car systems.

The complexity of the project carried out is clear because it involves starting from scratch in research and the use of a large number of knowledge of different subjects and the real demonstration of proper operation. It is noteworthy that despite develop a good theoretical basis of the problem to be solved, its proper functioning in practise presents additional difficulties with respect to the theoretical model developed.

The results have been satisfactory to achieve the proposed tasks from the beginning. The slot controller has a colour screen that provides an intuitive interface with the placement of the push buttons. The wireless communication allows more independence and different driving modes give to the controller new features never seen in previous slot car systems. The best reward is the development of a Master's Thesis project that has interest to the people.

PREFACE

The work reported in this Master of Science Thesis was conducted at the Department of Electronics and Communications Engineering in Tampere University of Technology, Finland during January 2015 – May 2015. This research is the result of collaboration Department of Electronics and Communications Engineering at Tampere University of Technology with Department of Electronics Technology at Technical University of Cartagena (Spain) under the Erasmus+ programme. The supervisor and the examiner of the thesis was Professor Jari Nurmi.

Firstly, I would like to thank Professor Jari Nurmi who acted as my supervisor for the Master's Thesis and proved to be a wonderful help to get me started with the study at Tampere University of Technology. I would like to extend my gratitude to Katja Laine for helping me with the electronics materials and the access to the laboratories. I would also like to thank my colleague Jose Antonio Martinez for helping me from Spain.

I also owe a big thank you to Laura, the architect of my happiness who stood by me despite the distance and endured my stress when I needed to let it all out. I would also like to thank my parents Ginés and María José, for the absolute support they have provided me throughout my studies. Thank you to my sister Úrsula and all my family for their motivation and for believing in me.

This thesis concludes an important time in my life. I would like to thank all my friends for making the years in university a beautiful stage of my life. Also thank you to everyone who has supported me throughout all my school years.

With gratitude, I owe this moment to all of you.

In Tampere, Finland, on 20th May 2015.

Santiago Ros Navarro

CONTENTS

1. INTRODUCTION 1
1.1 Project objectives 1
1.2 Project phases 2
2. STATE OF ART OF SLOT CAR RACING 4
2.1 Introduction 4
2.2 Historical overview 5
2.2.1 Slot car racing in Spain 7
2.2.2 Slot car racing in Finland 8
2.3 Technical aspects 9
2.3.1 Analog slot operating principle 9
2.3.2 Digital slot operating principle 10
2.4 Basic physical principles of slot car racing 11
2.5 Slot vehicles 15
2.5.1 Components 15
2.5.2 Common scales 19
2.6 Slot controllers 20
2.6.1 Analog controllers 20
2.6.2 Digital controllers 21
2.7 Slot tracks 21
2.7.1 Materials 21
2.7.2 Modifications 22
2.8 Competitions classes and rules 23
2.9 Hobbyists 24
2.10 Manufacturers 24
2.10.1 Carrera 24
2.10.2 Fly Slot 25
2.10.3 Ninco 25
2.10.4 Racer 25
2.10.5 Scalextric 25
3. ANALYSIS OF DIGITAL SLOT CAR SYSTEMS 27
3.1 Introduction 27
3.2 Digital slot car systems 27
3.2.1 Analog controllers 28
3.2.2 Digital controllers 32
3.3 Slot car system selected 35
3.3.1 Advantages and disadvantages 35
3.3.2 Ninco digital system 36
4. RESEARCH METHODOLOGY AND CONTROLLER FEATURES 41
4.1 Introduction 41

4.2 Research methodology 41
4.3 Research process 42
4.3.1 Test slot track 42
4.3.2 Connection 43
4.3.3 Basic slot controller diagram 43
4.3.4 Preview prototype 45
4.3.5 Final slot car controller 47
4.3.6 Improvements 49
5. HARDWARE ARCHITECTURE 51
5.1 Description of the hardware architecture 51
5.1.1 Transmitter module 52
5.1.2 Receiver module 52
5.2 Processing unit 52
5.2.1 Microcontroller 52
5.2.2 Selection of the microcontroller 53
5.2.3 Arduino platform 54
5.2.4 Arduino advantages 55
5.2.5 Arduino Nano 55
5.3 Digital-to-analog converter 59
5.3.1 Practical operation 59
5.3.2 Resistor ladder 60
5.3.3 Accuracy of resistor ladder 62
5.3.4 Buffer amplifier 62
5.4 Front panel 63
5.4.1 Display 64
5.4.2 Push buttons 67
5.4.3 Power source 68
5.4.4 Communications 71
6. SOFTWARE ARCHITECTURE 76
6.1 Introduction 76
6.2 Pin assignment 76
6.2.1 Transmitter module 77
6.2.2 Receiver module 77
6.3 Transmitter software 78
6.3.1 General structure 78
6.3.2 Global variables 79
6.3.3 Libraries 80
6.3.4 Welcome message 80
6.3.5 Instructions 82
6.3.6 Linear mode 83
6.3.7 Logarithmic mode 88
6.3.8 Kids mode 93

6.3.9 Automatic mode 95
6.4 Receiver software 98
6.4.1 General structure 99
6.4.2 Global variables 100
6.4.3 General lane change algorithm 100
6.4.4 Automatic lane change algorithm 101
6.4.5 Decimal to binary conversion 101
7. CONTROLLER IMPLEMENTATION 103
7.1 Introduction 103
7.2 Slotpiral R-evolution transmitter 103
7.2.1 Front panel design 104
7.2.2 Processing unit design 106
7.2.3 Power source design 107
7.2.4 Communications design 107
7.2.5 Final implementation 108
7.3 Slotpiral R-evolution receiver 108
7.3.1 Front panel design 109
7.3.2 Processing unit and digital-to analog converter design 111
7.3.3 Power source design 112
7.3.4 Communications design 113
7.3.5 Final implementation 114
8. ANALYSIS AND RESULTS 115
8.1 Analysis of Slotpiral R-evolution 115
8.1.1 Visual analysis 115
8.1.2 Laboratory tests 115
8.1.3 Race analysis 116
8.2 Cost report 117
8.2.1 Transmitter module cost report 117
8.2.2 Receiver module cost report 118
8.2.3 Total cost report 118
8.3 Discussion of the results 119
9. CONCLUSIONS 121
9.1 Final conclusion 121
9.2 Future works 122
REFERENCES 124

APPENDIX A: Source code of Slotpiral R-evolution transmitter 128

APPENDIX B: Source code of Slotpiral R-evolution receiver 142

APPENDIX C: Electronic schematics 143

LIST OF FIGURES

Figure 1. *Slot vehicle patent by Albert E. Cullen, 1936.....5*

Figure 2. *Maserati 250F (left) and Ferrari 375 slot cars by Minimodels.....6*

Figure 3. *Scalextric advertising poster in Spain, 1967.....7*

Figure 4. *How to make a slot car track, 1968.....8*

Figure 5. *Slot car track with typical electrical circuit.....9*

Figure 6. *Ninco digital control unit.....10*

Figure 7. *Force diagram of slot car mechanism.....11*

Figure 8. *Components of a slot car.....15*

Figure 9. *Types of slot vehicle transmission.....17*

Figure 10. *Belt and pulleys transmission system.....18*

Figure 11. *Models of the Ford GT, in 1/24, 1/32 y 1/64 scales.....19*

Figure 12. *Analog (left) and digital slot controllers.....20*

Figure 13. *Handmade wooden rally track, inspired by the Targa Florio.....22*

Figure 14. *8-lane speed track in Helsinki (Finland).....23*

Figure 15. *Types of slot car controllers.....28*

Figure 16. *Scalextric, the Digital System slot car controller.....30*

Figure 17. *Carrera Digital (left), Evolution and Wireless (right) slot car controllers..30*

Figure 18. *N-Digital (left) and Progressive slot car controllers.....31*

Figure 19. *Automatic or ghost slot car controller.....32*

Figure 20. *SCP-2 slot car controller, manufactured by Slot.it.....33*

Figure 21. *Scorpius Wireless controller.....34*

Figure 22. *Digital Mastertrack components.....36*

Figure 23. *Digital Starter Box.....36*

Figure 24. *Control Tower.....37*

Figure 25. *Pit Lane.....38*

Figure 26. *Simple (left), Double and Double Curve (right) Lane Changes.....38*

Figure 27. *Decoder chip and equipment.....39*

Figure 28. *Multi-Lane Sensor.....39*

Figure 29. *Ninco Multifunction Digital Control Unit.....40*

Figure 30. *Slot track used in laboratory tests.....43*

Figure 31. *4P4C connectors.....43*

Figure 32. *Basic diagram of Ninco slot controller.....44*

Figure 33. *Slotpiral front panel.....46*

Figure 34. *Slotpiral R-evolution transmitter (left) and receiver.....48*

Figure 35. *Transmitter (above) and receiver hardware architecture block diagram....51*

Figure 36. *Atmel microcontroller encapsulated in a 28-pin package.....53*

Figure 37. *Arduino Serial board.....54*

Figure 38. *Arduino Nano board front (left) and rear.....56*

Figure 39. *Arduino Nano pinout.....57*

Figure 40. *2 bits digital-to-analog converter.....60*

Figure 41. *Basic n-bit R-2R resistor ladder.....61*

Figure 42. *Ideal voltage buffer amplifier.....62*

Figure 43. *Thévenin source drives a unity gain voltage buffer.....63*

Figure 44. *Structure of thin-film-transistor liquid-crystal display.....64*

Figure 45. *Typical single SPI bus master and slave.....65*

Figure 46. *Arduino TFT LCD pinout.....66*

Figure 47. *Omron push buttons with square heads.....67*

Figure 48. *Active-low (left) and active-high push button connections.....68*

Figure 49. *Service hours of 9V battery with constant power.....69*

Figure 50. *Block diagram (left) and pin assignment of LM7810.....70*

Figure 51. *Unshielded twisted pair.....71*

Figure 52. *IEEE 802.15.4 protocol stack.....73*

Figure 53. *XBee Series 1 pinout.....74*

Figure 54. *UART data packet 0x1F as transmitted through the XBee.....74*

Figure 55. *System Data Flow Diagram in a UART.....75*

Figure 56. *General flowchart of the transmitter software.....78*

Figure 57.*Welcome message screenshot.....81*

Figure 58. *Instructions screenshot.....83*

Figure 59*. Flowchart of the linear mode.....84*

Figure 60. *Brake, pass and turbo actions screenshots in linear mode.....88*

Figure 61. *Flowchart of the logarithmic mode.....89*

Figure 62. *Brake, pass and turbo actions screenshots in logarithmic mode.....93*

Figure 63. *Flowchart of the kids mode.....94*

Figure 64. *Brake, pass and turbo actions screenshots in kids mode.....95*

Figure 65. *Flowchart of the automatic mode.....96*

Figure 66. *Screenshots in automatic mode at medium speed.....97*

Figure 67. *Transmission and reception of the variable num.....98*

Figure 68. *General flowchart of the receiver software.....99*

Figure 69. *Slotpiral R-evolution logo.....103*

Figure 70. *Block diagram of the hardware architecture of the transmitter module.....103*

Figure 71. *Front panel evolution (left to right).....104*

Figure 72. *Button L, central and button R printed circuit board masks.....105*

Figure 73. *Button L, central and button R printed circuit board with components.....106*

Figure 74. *Connections diagram of the transmitter processing unit.....107*

Figure 75. *Connections diagram of the XBee shield.....108*

Figure 76. *Connections of Slotpiral R-evolution transmitter module.....108*

Figure 77. *Block diagram of the hardware architecture of the receiver module.....109*

Figure 78. *PB shield (left) and DAC shield printed circuit board masks.....110*

Figure 79. *PB shield (left) and DAC shield printed circuit board with components....111*

Figure 80. *Digital-to-analog converter basic electrical diagram.....112*

Figure 81. *Connections diagram of the receiver processing unit.....112*

Figure 82. *Connections diagram of the wired communication.....113*

Figure 83. *Connections of Slotpiral R-evolution receiver module.....114*

Figure 84. *Oscilloscope screenshots of the laboratory tests.....116*

Figure 85. *UPCT experimental digital slot car system.....117*

Figure 86. *3D designs of Slotpiral R-evolution transmitter (left) and receiver.....119*

Figure 87. *Button L shield schematic.....143*

Figure 88. *Button R shield schematic.....143*

Figure 89. *Central shield schematic.....144*

Figure 90. *DAC shield schematic.....145*

Figure 91. *PB shield schematic.....146*

Figure 92. *XBee shield schematic.....146*

LIST OF TABLES

Table 1. *Braids properties by material.....16*

Table 2. *Arduino platforms basic specification.....56*

Table 3. *Types of 9V batteries.....69*

Table 4. *Unshielded twisted pair technical specifications.....71*

Table 5. *Specifications of the XBee Series 1.....73*

Table 6. *Pin assignment of transmitter module.....77*

Table 7. *Pin assignment of receiver module.....77*

Table 8. *Characteristics of the control variable in the linear mode.....83*

Table 9. *Characteristics of the control variable in the logarithmic mode.....90*

Table 10. *Characteristics of the control variable in the kids mode.....95*

Table 11. *Characteristics of the control variable in the automatic mode.....98*

Table 12. *Transmitter module cost report.....117*

Table 13. *Receiver module cost report.....118*

Table 14. *Total cost report.....118*

LIST OF SYMBOLS AND ABBREVIATIONS

3D	Tree Dimensional
AC	Alternating Current
DC	Direct Current
DIP	Dual In-line Package
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPROM	Erasable Programmable Read-Only Memory
FTDI	Future Technology Devices International
I / O	Input / Output
IEEE	Institute of Electrical and Electronics Engineers
ITO	Indium Tin Oxide
LCD	Liquid-Crystal Display
LED	Light-Emitting Diode
LR-WPAN	Low-Rate Wireless Personal Area Network
PAN	Personal Area Network
PECVD	Plasma-Enhanced Chemical Vapor Deposition
RAM	Random-Access Memory
RGB	Red Green Blue
ROM	Read-Only Memory
rpm	Revolutions per minute
Rx	Reception
SPI	Serial Peripheral Interface
TFT	Thin-Film Transistor
TTL	Transistor–transistor logic
TUT	Tampere University of Technology
Tx	Transmission
UART	Universal Asynchronous Receiver-Transmitter
UPCT	Universidad Politécnica de Cartagena, Technical University of Cartagena
USB	Universal Serial Bus

1. INTRODUCTION

In this introductory chapter, the project objectives and the different chapters in which it has been divided are presented. Objectives describe what has been developed and the overall progress of the project is justified. It also analyzes the complexity that can entail and what is expected to achieve completion.

The chapters describe in a clear and concise manner the various aspects which have been the subject of research and development. Starting with the first phase of the study of the state of the art up to the last, where possible conclusions and future works are presented. The various objectives achieved in each chapter are exposed in a way that is useful both the author of the project as to any person who read this Master's Thesis to know the process of elaboration followed.

1.1 Project objectives

Throughout this project, the research is developed around a model of digital slot car circuit marketed by the Spanish company *Ninco*, equipped with a multifunction digital control unit which is the heart of the system and it is responsible for managing the speed of slot vehicles by the different lanes of a slot car circuit.

The objectives of this project are the design, development and implementation of an innovative electronic controller for digital slot car applications that allows more precise control of slot vehicles as well as new features that do it more intuitive. The work is divided into two main phases: one dedicated to the development of electronic hardware prototype and another focused on the development of software control interface that allows the precise adjustment of speed and better handling on track.

For the implementation of this project, a low-cost microcontroller device is used and programmed with the developed control software. It has also been equipped with conditioning electronics required for a proper operation of the slot controller.

The most common components of digital slot control units on the market have been analyzed for the development of this project and it contains a design proposal that implements a low cost microcontroller that provides intelligence and advanced control to slot systems.

This project ends with the complete implementation of an innovative digital slot controller with wireless connection, different driving modes and an intuitive interface that

includes a TFT LCD screen. Furthermore, this project is the theoretical basis for another Master's Thesis which is being carried out at the Technical University of Cartagena (UPCT), in Spain.

1.2 Project phases

This current project is structured in certain phases that correspond to the following chapters:

- **Chapter 2. State of art of slot car racing.** First, an investigation has been conducted on the most significant aspects that encompasses the world of slot car racing and the most important characteristics of this are highlighted. The chapter begins by performing a global historical overview of slot car racing and then it focuses on the evolution that has experienced in Spain and Finland. Technical aspects of operation are also analyzed, differentiating between digital and analog nature, in addition to deepen in the physical principles of greater importance for slot car racing. Then, the main elements involved in the world of slot are studied, such as slot vehicles and components, analog and digital controllers, along with the structure of this hobby and different circuits or tracks. Topics such as the competitions, the official rules and a brief description of slot fans types are also addressed. Finally, an analysis of the leading slot car system manufacturers worldwide is performed.

- **Chapter 3. Analysis of digital slot car systems.** This chapter presents a thorough analysis of the different slot car controllers available today. All aspects related for digital slot car systems are addressed, starting with an overview of the analog slot car controllers made by amateurs, continuing with commercial slot car controllers that develop the main specialized manufacturers and making a research of the latest advances in digital slot car controllers. Digital slot car system used in this project marketed by the Spanish company *Ninco* is also presented, showing its main characteristics, tracks, accessories and the digital control unit. In addition, a comparison of selected slot car system is performed compared to other commercially available, detailing its advantages and disadvantages.

- **Chapter 4. Research methodology and controller features.** Firstly, this chapter explores the methodology followed for the development of the slot controller, indicating the stages of analysis, design and implementation. The research process of *Ninco* slot car system is also presented because there is not reliable information about it in the official references. In addition, the most significant laboratory tests for the correct evaluation of alternatives are shown, obtaining quantitative data of great importance. This chapter also aims to show the features included in the developed slot controller, describing the improvements im-

plemented and highlighting its characteristics compared to other controllers available on the market, which have been discussed in the previous chapter.

- **Chapter 5. Hardware architecture.** This section describes the hardware architecture of the slot controller and the salient elements of the processing system such as digital - analog conversion, graphical interface, front panel, power source and communication system used. Subsequently, a detailed theoretical study of the features presented in Chapter 4 having regard to principles, characteristics and types available for project implementation is done. First, an overview of possible devices that can be used is presented to finish with the definitive devices finally selected.

- **Chapter 6. Software architecture.** A detailed analysis of the software code used in the low cost microcontroller that contains the slot controller is conducted in this chapter. The source code and control algorithms used in the software are exhaustively explained through flowcharts and the main characteristics of driving modes of the slot controller are specified by comparative tables representing the motor power bands of slot vehicles.

- **Chapter 7. Controller implementation.** In this chapter, the implementations of the different functional blocks that are part of the slot controller are shown in detail by assembly diagrams of the manufacturing process. The evolution of electronic circuit designs, debugging and optimization, besides the final assembly process are also shown. Furthermore, the wireless communication system comprising the transmitter module and receiver module is exposed.

- **Chapter 8. Results and analysis.** Once the design process, with the final implementation of the slot controller, is completed, some series of tests and experiments that prove its correct operation were performed. This chapter presents the different experiments which the slot controller was subjected to prove its proper operation. In addition, an economic study with the costs of each component used to implement the slot controller is included here. Finally, the results obtained after the development of the project are discussed.

- **Chapter 9. Conclusions.** In the last chapter, the conclusions reached at the end of the project are set out and the most relevant problems encountered during the implementation thereof are cited. Finally, there is also a section where potential projects related to research line developed in this project are proposed.

2. STATE OF ART OF SLOT CAR RACING

In this chapter, an investigation on the most significant aspects that encompasses the world of slot car racing has been conducted and the most important aspects of this are highlighted.

The chapter begins by performing a global historical overview of slot car racing and then it focuses on the evolution that has underwent in Spain and Finland. Technical aspects of operation are also analyzed, differentiating between digital and analog slot, in addition to deepen in the physical principles of greater importance for slot car racing.

Then, the main elements involved in the world of slot are studied, such as slot vehicles and components, analog and digital controllers, along with the structure of this hobby and different circuits or tracks. Topics such as the competitions, the official rules and a brief description of slot fans types are also addressed.

Finally, an analysis of the leading slot car system manufacturers worldwide is performed.

2.1 Introduction

Slot car racing or also called slot racing is a competitive hobby of racing with powered miniature vehicles equipped with electric motors which are guided by slots or grooves in a special track or circuit.

The minimum materials needed for the slot car racing are a slot car, the track, a power supply to generate electricity and a controller for the slot car. The main difficulty of slot racing is to achieve the fastest vehicle speed possible without leaving the track. The circuits include powerful magnets that keep the chassis of slot vehicles closest to the tracks to facilitate the piloting.

The word “slot” comes from the English language in reference to the grooves through which flows the guide flag of slot automobile and takes the current to power its electric motor. Although the term slot is the most correct, in some countries this hobby is better known by the name of the manufacturer *Scalextric* [1].

This hobby has existed since 1952 but it has not always enjoyed the same fame and acceptance. Today slot racing is diversified according to the scale of vehicles. The more common in Europe and South America is the 1/32 scale, depending on the country it

also highlight the 1/24 and 1/43, particularly in United States of America. The 1/83 scale is used in some cases when the space available for the track is limited.

2.2 Historical overview

The first slot cars were marketed by American toymaker *Lionel Corporation* since 1912 and these vehicles worked with electricity supplied from train groove, similar to the modern slot, but the production was stopped in 1915. Similar systems appeared sporadically during the next four decades. The first slot car was patented in March 1936 by Albert E. Cullen (Figure 1), but until the end of 1950 the majority of toy vehicles were guided by elevated rail similar to trains.

At the end of the 1930s, the cars used were of relatively large scales, such as 1/16 or 1/18, equipped with small two-stroke engines with spark plugs. These vehicles were fastened to the center rail of a circular track, called “tether car” in the United Kingdom and there was not a human control over the car, so it was a hobby with mechanical interest [3].

In the 40s, the British modelers began experimenting with cars driven by electric artisanal motors and in the 50s cars used the same engines as the miniature electric trains. The company *Minimodels Ltd.*, based in London and dedicated to the manufacture of metal toys, began manufacturing tin vehicles from 1947. The British engineer Fred Francis, director of the company decided to include a mechanism to give movement to his reproductions of cars in 1952 [4].

In 1954, the British *Southport Model Engineering Society* was claimed by the patent owner for exhibitions and get donations of money, so the members built an electric racing track with 6 lanes and 18 meters length for 1/32 car models, which are considered as the progenitors of the current slot car racing.

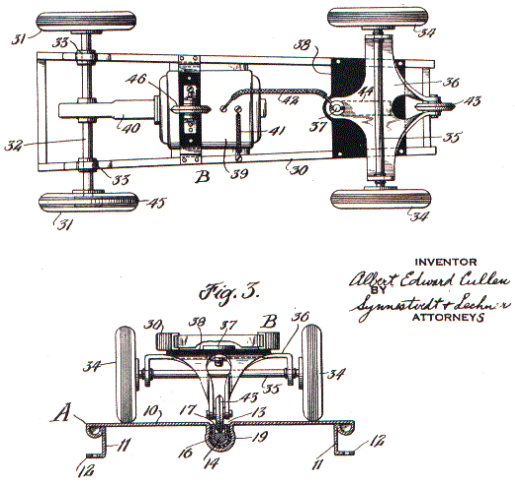

Figure 1. *Slot vehicle patent by Albert E. Cullen, 1936* [2].

Several clubs in the United States of America built similar tracks with rails on the track surface to mid-50s. The term "slot car" was used to differentiate them from the first cars, which were moved by common rail surface. So many clubs proliferated with tracks made by fans. There were many advantages of the new system grooved surface opposite the old rails and clubs were favored more and more by the new slot system [5].

Minimodels Ltd. manufactured seven models of clockwork vehicles between 1952 and 1956, called "Scalex". The first cars manufactured were Ferrari 375 Grand Prix and Maserati 250F Grand Prix (Figure 2). After the good reception by the public, the company improved the clockwork mechanism creating a new range of cars called "Startex". After a decline in sales, Fred Francis added an innovation to its "Startex" models, a small electric motor which represented a growth in sales. In 1957, slot car racing packs were sold with all accessories. The success was so high that the company could not produce all the demand and ended with the purchase of *Minimodels Ltd.* by *Lines Bros Ltd*, also known as *Tri-ang* in 1958. Throughout 1958 and 1959, *Scalextric* models were edited using cars "Scalex", but adapted to run on the tracks [6].

In 1960, plastics began to expand in industry and *Scalextric* also used this material because it allowed further detailed models. The British Formula One Lotus 16 was the first slot vehicle manufactured with plastic, followed by Vanwall, Lister Jaguar and Aston Martin DBR [8].

Scalextric became a worldwide success in 1964 and its products were sold in France, Spain, Australia, New Zealand and the United States of America. The magnitude of this success was so great that the Formula One driver Jim Clark was responsible for promoting the brand throughout the world [9].

In the 70s, slot car racing was already present in many homes and the first slot associations at national and international level were created to establish rules for competitions. Technological innovations increased the speed of the cars in all scales and improved engines, tires and magnets. The older fans did not like this new slot racing because it was highly specialized and they preferred the original slot although it was slower.

Figure 2. *Maserati 250F (left) and Ferrari 375 slot cars by Minimodels* [7].

In the 90s, computer design and 3D printing allowed the creation of more detailed car models, so high competition parts for slot cars began to be manufactured.

In 2004, a new system called “slot digital” appeared on the market offering new characteristics for the 1/32 scale, such as the ability to compete with more cars on the track, timing control by digital control unit and possibility of overtaking [10].

2.2.1 Slot car racing in Spain

In 1962, the Spanish company *Exclusivas Industriales S.A.* also known as *Exin* reached a trade agreement of exclusivity with the British *Lines Brothers Ltd.* to distribute and manufacture its most successful slot cars. These vehicles were 1/32 scale and they circulated on grooved circuits equipped with electric motors. *Scalextric* was originated in Spain and its head office was installed in Barcelona, where it was officially unveiled at Barcelona Toy Exhibition [11].

Initially, the reception of public and industry professionals was not expected and it could be described as indifferent. *Scalextric* had disadvantages because its name was a little difficult to pronounce in Spanish and it was a toy without fame yet. Even so *Exin* overcame this obstacle with its team of publicists and promoters, which quickly got to *Scalextric* became the favorite Christmas toy for most children of the time.

Figure 3. *Scalextric advertising poster in Spain, 1967* [12].

The economic crisis had devastating effects in the mid-80s and forced it to close enough companies. *Exin* also had poor economic results and this added to the growing market of video games did the company finally closed in 1993, after 31 years making toys. The American company *Tyco Toys* acquired the license to market *Scalextric* and was responsible for maintaining the production of slot cars. Manufacturing was moved to China, which also affected the decrease of the quality of the cars [13].

Following the closure of *Exin* a new company was created by two of its former employees. So in 1993, Eduard Nin and Eladio Cosculluela founded *Ninco Desarrollos S.L.*, commonly called *Ninco*.

The first project of the company was a scaled replica of the Renault Clio 16V. This slot car broke all the standards of the time, setting its goal as the amateur hobbyist, adult and expert. The success of the range of vehicles that followed this first model was completed in 1997 with the appearance of *Ninco* tracks. By their design, performance and ease of use, they were considered the best slot tracks in the world [14].

In 2006, *Ninco* added to its activities a distribution company in Spain of international hobby brands. Since 2009, it has been developing and marketing its own brands of radio control. *Ninco* is the Spanish leading manufacturer of slot cars and tracks and it distributes products for the entertainment of children and adults in over 30 countries worldwide.

2.2.2 Slot car racing in Finland

Slot car racing has existed in Finland since early 60s and it had high peak during the early years. The record of highest attendance in Finnish Championships was in 1964, with over 300 entries [15]. In 1968, slot hobby was so famous in Finland that a video explaining how to make an artisanal slot track was aired on television (Figure 4).

Figure 4. *How to make a slot car track, 1968* [16].

Nowadays, slot car racing is less famous, about 30 to 100 entries in major races but clubs activity is quite good. In Finland there are not commercial slot circuits, all tracks are in clubs. These clubs are owned by the community, church or motor racing club, so it costs virtually nothing to be a member. Probably this is why our average racers are fairly young. In addition, classes allow younger drivers to compete in their own categories. Finnish slot racing is strictly 1/24 scale and the tracks are fast style, banked 8 or 6 lane tracks although there are only a few enthusiasts who do some 1/32 scale racing.

2.3 Technical aspects

Then the principles of operation of analog and digital slot systems are described, highlighting the features that differentiate them.

2.3.1 Analog slot operating principle

The operation principle of analog slot system is based on the following points described below:

- Electricity for the slot vehicle's motor is carried by metal strips next to the groove, and it is picked up by two electrical contacts of conductive material such as copper braid. These braids are mounted alongside a swiveling blade or guide flag under the front of the slot car.
- The voltage of the slot car is varied by a resistor or rheostat in the controller.

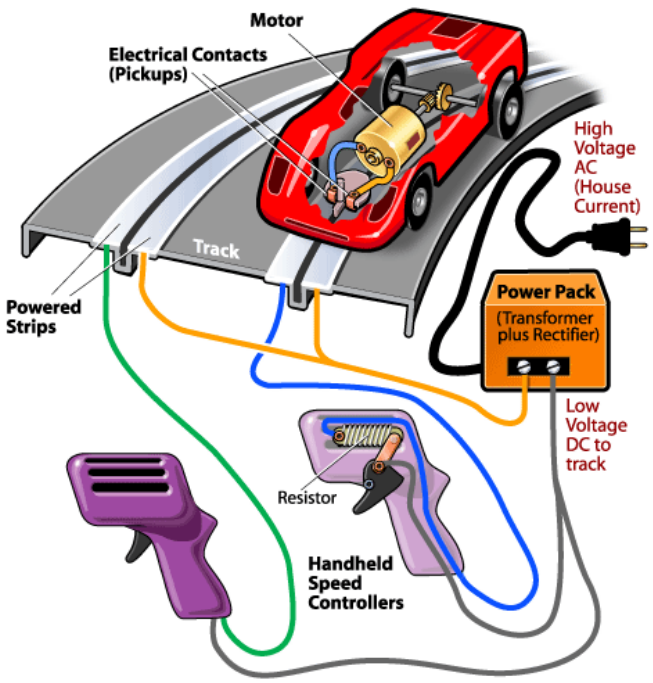

***Figure 5. S**lot car track with typical electrical circuit* [17].

- An external current transformer is responsible for converting alternating current (AC) from the mains to direct current (DC) that is used to power the circuit.
- Analog slot systems only allow one vehicle in each lane, so if the track has two lanes it uses two controllers.
- The slot vehicles are equipped with magnets under the car to improve the adherence to the track. Professional fans prefer to remove the magnets to have more fun, because it allows greater human control of the vehicle. In Spain there are few competitions that allow the use of magnets but only junior classes are authorized to use them.

2.3.2 Digital slot operating principle

The operation principle of digital slot system is similar to the analog system previously explained, with some differences as indicated below:

- The control of digital slot cars is via digital signal and it allows several vehicles in the same lane, because each car carries a single decoder chip that identifies it.
- Communications are done via a digital control unit that sends unique data for each car and powers the lanes of the track. The control unit (Figure 6) also allows multiple options, such as measuring times, different game modes and shows the overall ranking.
- Most common tracks have two lanes which are fed by the digital control unit that receives power from an AC / DC converter, as the analog version, but this has sections where the rails cross to allow change lanes to overtake.
- Controllers have a rheostat to vary the voltage supplied to the track and include a button to pass, which occurs when it is pushed.
- Analog slot vehicles are not compatible with digital slot because analog cars are not equipped with the appropriate electronics to decode the combination of voltage and information produced by the multifunction digital control unit [18].

Figure 6. *Multifunction digital control unit*[1].

2.4 Basic physical principles of slot car racing

The theoretical system studied is ideal, the friction between all moving parts and moments of inertia of rotating parts are negligible. Tire adhesion to the ground is perfect, the rotational movement transmitted from the wheels is without slipping, so assuming infinite adhesion.

Track is considered flat and perpendicular to the Earth's radius, the gravity does not affect the movement of the vehicle and the frictional force is negligible. There are established laws that relate the magnitudes of torque, force, acceleration, work and power in slot vehicles. Basic physical force diagram involved in slot car racing and mechanism are discussed below (Figure 7).

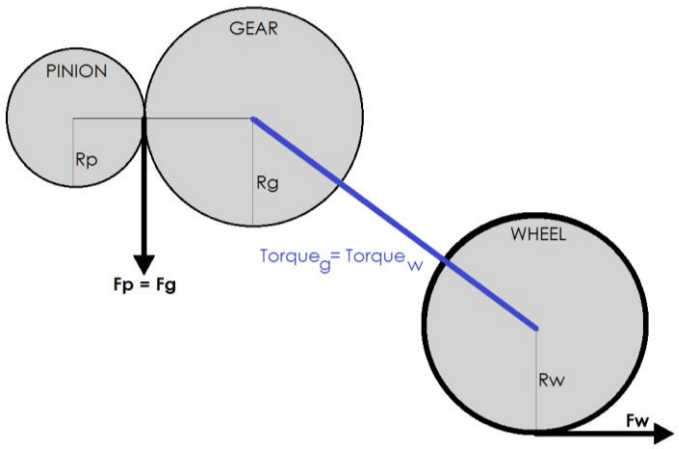

Figure 7. *Force diagram of slot car mechanism*[2].

Slot car engine rotation is converted into translational motion of the vehicle and some of the parameters involved are:

The motor rotates about its longitudinal axis with a constant angular velocity producing a driving torque. Associated to the motor shaft is attached a pinion with radius R_p that share its angular momentum or torque with the motor torque so the pinion torque is equal to the motor torque:

$$T_p = T_m \tag{1}$$

where T_p denotes the torque of the pinion and T_m means motor torque.

Perfectly engaged in the pinion and with equal number of teeth per unit length, there is a gear with radius R_g which rotates driven by the tangential force exerted on the pinion and it generates a torque on the rotation axis of the gear.

$$F_p = F_g \tag{2}$$

where F_p denotes the force of the pinion and F_g means gear force.

Along the axis of the gear is coupled the wheel with radius R_w, for this reason the gear torque is the same that the wheel torque:

$$T_w = T_g \tag{3}$$

where T_w denotes the torque of the wheel and T_g means gear torque.

Consequence of angular momentum in section perimeter of the wheel that rubs against the track, the wheel tangential force is exerted. This is the force which pushes back on track.

As stated the famous law of action-reaction or third Law of Newton:

"To every action there is always opposed an equal reaction: or the mutual actions of two bodies upon each other are always equal, and directed to contrary parts".

Therefore, if the wheel exerts a force (F_w) on the track, the track also exerts a force but opposite on the wheel and the vehicle ($-F_w$). The negative sign is omitted because the force is absolute value and finally the force that propels the slot vehicle is F_w.

Based on the following classical mechanics equations:

$$Torque = F * R \tag{4}$$

where F denotes the tangential force to the rotation and R means the distance to the rotation axis.

$$F = m * a \tag{5}$$

where F denotes the force used to accelerate the mass m and a means the resultant acceleration.

Expressing F_w depending on the torque and substituting equation (4) on each of the elements forming the set pinion-gear-wheel:

$$T_p = F_p * R_p \tag{6}$$

$$T_g = F_g * R_c \tag{7}$$

$$T_w = F_w * R_w \tag{8}$$

where T_i denotes the torque of each element, F_i denotes the force of each element and R_i means the distance to the rotation axis of each element.

Solving forces:

$$F_p = {T_p}/{R_p} \tag{9}$$

$$F_g = {T_g}/{R_g} \tag{10}$$

$$F_w = {T_w}/{R_w} \tag{11}$$

From equation (11) and replacing it with the equations presented above:

$$F_w = \frac{T_w}{R_w} = \frac{T_g}{R_w} = \frac{F_g * R_g}{R_w} = \frac{F_p * R_g}{R_w} = \frac{\frac{T_p}{R_p} * R_g}{R_w} = \frac{T_m * R_g}{R_w * R_p} \tag{12}$$

Using equation (5):

$$F_w = m \cdot a = \frac{T_m \cdot R_g}{R_w \cdot R_p} \tag{13}$$

Solving the acceleration:

$$a = \frac{T_m \cdot R_g}{R_w \cdot R_p \cdot m} \tag{14}$$

Finally, this theoretical and reasoned explanation of the alleged premises is obtained:

"Acceleration is directly proportional to torque and gear radius and inversely proportional to the pinion radius, the wheel radius and the total vehicle mass".

Thereupon, the physical quantities of work and power are analyzed, employing the following classical mechanics equations:

$$W = F \cdot S \tag{15}$$

where W denotes work, F means force and S means space.

$$P = W/t \tag{16}$$

where P denotes power, W means work and t means time.

The work of the wheel (W_w) is calculated by the distance traveled in one complete revolution, with negligible slip. This space coincides with the length of the circumference of the wheel:

$$S_w = 2\pi \cdot R_w \tag{17}$$

where S_w denotes the distance traveled in one complete revolution and R_w means radium of the wheel.

Substituting equation obtained in the previous equation (15):

$$W_w = F_w \cdot S_w = \frac{T_m \cdot R_g}{R_p \cdot R_w} \cdot 2\pi \cdot R_w = \boldsymbol{\frac{2\pi \cdot T_m \cdot R_g}{R_p}} \tag{18}$$

As can be seen after simplification, the radius of the wheel (R_w) does not influence the work.

Time taken by the wheel in a full turn is also necessary to know. It is determined by the speed of the motor and the reduction ratio of pinion-gear set:

$$W_m \cdot Pinion\ teeth = W_w \cdot Gear\ teeth \tag{19}$$

where W_m denotes the motor angular velocity in revolutions per second (rps) and W_w means wheel rotational speed in revolutions per second (rps).

Pinion and gear must have the same number of teeth per unit length to properly engage. Expressing the number of teeth depending on the length of the circumferences:

$$W_m \cdot 2\pi \cdot R_p = W_w \cdot 2\pi \cdot R_g \tag{20}$$

Solving W_w:

$$W_w = W_m \cdot \frac{R_p}{R_c} \tag{21}$$

Time is inverse of the rotational speed:

$$t = \frac{1}{W_w} = \frac{1}{W_m \cdot \frac{R_p}{R_g}} = \boldsymbol{\frac{R_g}{W_m \cdot R_p}} \tag{22}$$

Substituting equations (18) y (22) in the equation for power:

$$P_w = \frac{\frac{2\pi \cdot T_m \cdot R_{cg}}{R_p}}{\frac{R_g}{W_m \cdot R_p}} = \frac{2\pi \cdot T_m \cdot R_g \cdot W_m \cdot R_p}{R_p \cdot R_g} = \boldsymbol{2\pi \cdot T_m \cdot W_m} \tag{23}$$

where P_w denotes the power developed in one complete revolution by the torque T_m and with angular velocity W_m.

In conclusion, it is clarified that power is not related to the assembly of the vehicle transmission and only involves two factors to consider: torque and speed.

2.5 Slot vehicles

Cars are the main attraction of slot racing. There are real car models scaled from different eras and competitions. Among the different car brands available in the market, there are many models with different mechanical configurations but with common elements that are addressed in the following points.

2.5.1 Components

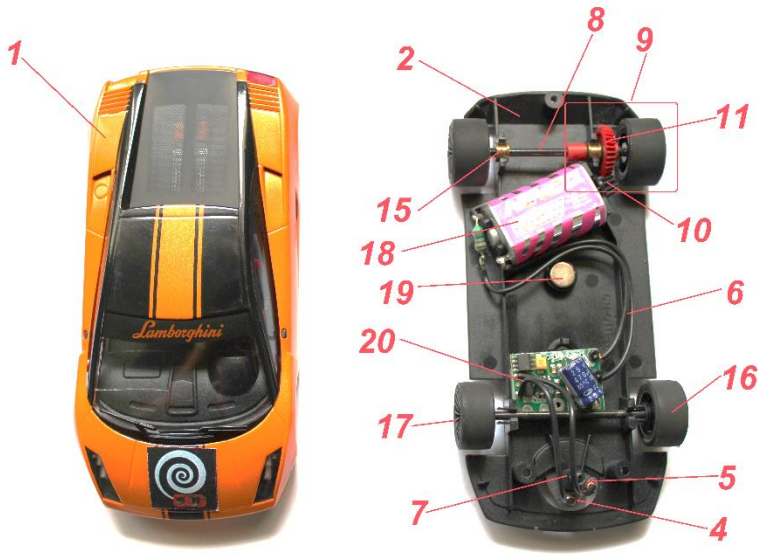

Figure 8. *Components of a slot car*[1].

- **1. Bodywork**: it has always been an element of study and improvement by slot manufacturers and fans. Bodyworks are manufactured in many materials, but the most commonly used are plastics such as acrylonitrile butadiene styrene (ABS) and acetate. Resin, fiberglass or polycarbonate resin are also used.

  Some bodyworks are removable and inside feature LED lighting system, driver, spare wheel, fire extinguishers, roll cage and other decorative elements. Vehicles usually have a driver and a co-driver in rally competitions, and speed competitions only take a driver. Metal or plastic screws are responsible for securing the bodywork to the car chassis. Most competitive hobbyist often use unpainted bodyworks for their vehicles weighing less.

- **2. Chassis**: consists of an internal framework made of plastic, steel, aluminum, titanium, fiberglass or carbon fiber that supports all the elements of slot car, such as motor, bearings, wheels and shafts.
- **3. Frame**: it is a subdivision of the chassis where the motor is located to allow better regulations. It is not present on all chassis and it may be made of plastic or metal.
- **4. Guide flag**: it is the only contact point with the track besides wheels. The guide flag directs the slot vehicle and receives electricity to the motor by two electrical contacts of conductive material such as copper braid. It is made of plastic or graphite on 1/24 scale.
- **5. Braids**: metallic contacts responsible for transmitting power from the track to the slot vehicle. Braids are made of copper, tin or silver (Table 1). Depending on the material from which are made, duration, driving and hardness can vary [19].

Table 1. *Braids properties by material.*

Material	Electrical conductivity	Durability	Rigidity
Cu	Low	High	Low
Ag	High	Low	High
Sn	Medium	Medium	Medium

- **6. Wiring**: conductor wires are responsible for transmitting electricity to the motor. There are different types but the most common are plastic coated copper. There are also silicone coated wires which can be molded. Some older slot cars have cooper strips instead of wires, but its main function is the same.
- **7. Tilting arm**: rally slot cars have an arm in the guide flag that allows tilt vertically to overcome obstacles. The tilting arm consists of a plastic or steel sprint.
- **8. Shaft**: it is a mechanical component for transmitting torque and rotation to the wheels. From the beginning shafts have been further developed by all manufacturers, to make the vehicle look more realistic model and improve performance. They are usually made of plastic or metal but there is a difference between them. Plastic material is used to manufacture the front shaft which does not require high strength. Therefore, lighter weight is favorable. The rear shaft or drive shaft is made of metals such as brass, iron, titanium, steel or aluminum. There are also threaded shafts that can regulate the distance between tires.
- **9. Transmission**: there are different arrangements of transmissions (Figure 9) listed below.
    - **Pancake**: this transmission arrangement has a flat commutator and vertical shaft. The power is carried by a chain of spur gears along the top of the chassis to a pinion which drove a crown gear at the axle [20].

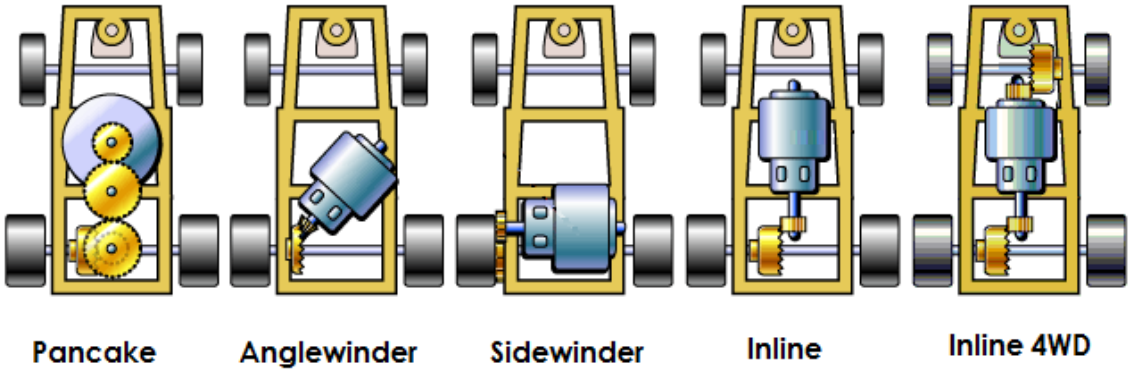

Figure 9. *Types of slot vehicle transmission* [23] [24].

- **Anglewinder**: this transmission arrangement has a motor which runs at an angle to the drive shaft and drives it through a bevel or other angled gear arrangement. It is a development of the sidewinder arrangement.
- **Sidewinder**: this transmission arrangement of the motor is parallel to the driven axle and power is transmitted through spur gears or a belt, friction or even by direct drive. The word also refers to the transversely-mounted motor of such a car [21].
- **Inline**: this transmission arrangement of the motor runs lengthwise down the chassis, perpendicular to the drive shaft. Power is transmitted through a pinion to a crown gear on the shaft, or through bevel gears. The word also refers to the longitudinally-mounted motor or the motor arrangement of such a car [22].
- **Inline 4WD**: this transmission arrangement of the motor transmitted power through two pinions to a crown gear on the front axle and another crown gear on the rear axle. These vehicles have four-wheel drive.

- **10. Pinion**: it is the primary gear that transmits power to the gear and it is made of materials such as nylon, bronze, brass or steel. Number of teeth varies from 8 to 13, allowing greater acceleration with the lower and higher speed with larger. The pinion is a fundamental part of the transmission together engine and gear. The biggest pinions provide more speed to slot cars but acceleration and vehicle stability are decreased.
- **11. Gear**: it is the middle gear that transmits power between the pinion and the drive shaft and is made of plastic materials. The core is made of aluminum or bronze. Number of teeth varies between 23 and 40 generally and it should be lubricated. The biggest gears provide more acceleration to slot cars but speed is decreased. A gear variant is the differential, a costly and delicate element that prevents the car from skidding.
- **12. Pulleys**: some slot car include pulleys for transmitting power to the rear to the front axle, but this may also be performed with another pinion and the inline 4WD arrangement explained before. Pulleys are usually made of aluminum or plastic.

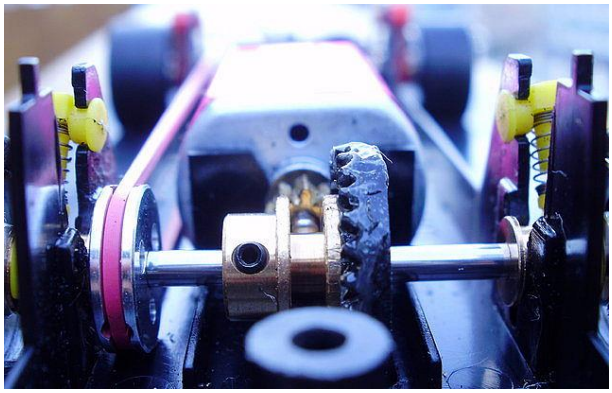

Figure 10. *Belt and pulleys transmission system*[1].

- **13. Belt**: it is a loop of flexible material used to link mechanically two pulleys. Belts are made of rubber and are differentiated by their length and section. A short belt transmits more power to the front shaft and improves braking, but has an abrupt behavior.
- **14. Suspension**: it is not characteristic of all slot car models but is used in rally vehicles. There are different shock absorbers depending on track type.
- **15. Bearing**: it is the fastening of the shaft and must be lubricated for correct operation of the car. Bearings are manufactured in various materials. Polytetrafluoroethylene (PTFE) is the best material because its friction is low. Bronze is also used because is cheaper. The ball bearings are not allowed by many competitions regulations.
- **16. Wheels**: they are responsible for transmitting motor power to the track. A few years ago, tire choice was limited to three options: smooth, hard or studded wheels, made of conventional natural rubber. Due to new method of injected and processing tires, there are available other compounds, such as silicone or vulcanized rubber which give more grip. Front wheels must have the least possible grip to avoid understeer (vehicle tends to escape to the outside of the curve) and excessive grip of the rear wheels can cause overturning.
- **17. Rim**: it is the support of the wheel. Rims must be securely fastened to the shaft and are usually constructed of plastic, although materials such as aluminum, titanium or nylon are also used. Cars with small rims have the center of gravity closer to the ground, so the slot car has more stability and acceleration, but less speed. Cars with larger wheels are less stable, but faster.
- **18. Motor**: it is responsible for producing movement of the vehicle and it is held by integrated brackets on the chassis. Formerly, each manufacturer produced its own motor with intrinsic characteristics. *Scalextric* motors were very famous for their quality and mechanical properties. With the emergence of new manufacturers to the slot sector, such as the Spanish *Ninco*, most companies began using a common denominator in their engines: the Japanese *Mabuchi Motors*. *Scalextric* motors reach 18 000 rpm, while Japanese engines are able to exceed 30 000 rpm. Motors have a very high level of degradation, not being effective in high

competition for more than 6 or 8 months and extending its life about 4 or 5 years for active users.

- **19. Magnet**: small piece that has a significant magnetic field that adheres to the surface of the vehicle frame and achieves greater speed, increased adhesion and avoids running out of the track. Magnets can be made of ferrite or neodymium which has greater magnetic power.
- **20. Decoder chip**: in Digital slot systems, vehicles are equipped with a small circuit that performs the task of decoding and analyzing the information it receives, which encodes data and power in the same signal. It also serves to identify each slot vehicle with its corresponding controller. The chip is connected in series between the guide and the car engine.

2.5.2 Common scales

These slot car scales are currently produced by manufacturers and the most used for slot competitions are:

- **1/24 scale**: it is the largest slot car scale used. Approximately, a typical 1/24 slot vehicle might be 18-20 cm long, so more space is required to install the tracks. This scale is demanded by professional slot racing clubs.
- **1/32 scale**: it is the most commercialized scale for home and amateur hobbyists. These slot vehicles are smaller than 1/24 scale, dimension can vary from 13 to 15 cm long.
- **1/43 scale**: it is based on the size of collection model cars and children's toys. This scale was first commercialized in 2006 and is gaining some acceptance among adult hobbyists for its affordability and moderate space requirements compared to the previous scales. The average size of the vehicles on this scale is 11 cm.

Figure 11. *Models of the Ford GT, in 1/24, 1/32 y 1/64 scales* [25].

- **1/64 scale**: it is also called HO scale has it is based on the size of model trains used in model railroading [26]. The dimension of model cars in this scale ranges between 5.5 and 8 cm. The main advantage is its small size because it does not require large premises to install the track [27].

2.6 Slot controllers

Slot controllers can be classified into two big groups. Thanks to the advancement of technologies are becoming increasingly popular digital controllers, as opposed to traditional analog controllers.

Figure 12. *Analog (left) and digital slot controllers*[1].

2.6.1 Analog controllers

The principle of operation of analog controllers is based on a simple variable resistor or rheostat, which manages the electric power by a cursor or sliding part that slides through the trigger. The resistance of the rheostat usually varies between 10Ω and 60Ω for analog slot systems. Better sensitivity is obtained with high resistance values.

Most popular slot controllers feature brake position, which helps stop to the vehicle quickly. This position causes an effect of electromagnetic brake on the motor when the throttle trigger is released. The result varies sometimes with a reducing brake system which limits braking power and consists of a variable resistor coupled to the brake cable.

These controllers can be purchased and assembled in stores or it can also be manufactured using traditional methods with the right skills. Analog controllers for digital slot systems are also equipped with a button for overtaking on special track sections enabled for it.

2.6.2 Digital controllers

Advances in slot controllers show their further development with digital controllers, which have become very popular in recent years. Highlighting some controllers, such as diodes controllers that base their operation principle on the serial association of diodes or control pulses.

Digital controllers are more accurate than analog, making it a very useful tool. The main disadvantages are its high price and fragility compared to the previous. These controllers have multiple regulations such as power control, acceleration or braking sensitivity.

As a novelty, highlighting controllers based on digital electronics, which also include the functions mentioned above, and provide better features like visual interface with display included.

2.7 Slot tracks

Slot tracks can be made from molded plastic commercial track sections or handmade using traditional methods. Circuits can be manufactured by different materials, sizes, number of lanes.

Commercial track sections allow the recreation of any imaginable track with the available space as limit. These commercial tracks recreate circuits formed by 1, 2, 4, 6 or 8 lanes, which can be divided into open, closed, rally, or speed circuits.

On the one hand, digital tracks typically use 2 lanes, although it is possible to incorporate up to 8 lanes. In this type of tracks, overtaking can be done in the track sections equipped to them and more slot vehicles can be driven per lane. Digital slot is not as well-known as the analog slot, due to its complexity and higher price.

On the other hand, there are handcrafted tracks. These slot tracks are not restricted lanes, because they are independent of prefabricated sections and may include slopes or obstacles on the track.

2.7.1 Materials

First slot circuit dating back to 1910 was made with natural rubber but their high maintenance cost made them stop using. At present, two types of materials are used exclusively:

- **Plastic**: tracks of this material are the most used and manufactured since 1950, when the company *Minimodels Ltd*. introduced them [28]. Typically high density polyethylene molding is used for manufacturing. Plastic tracks are inexpensive and easy to work with and the design of the circuit can be easily changed.

The joints between the sections, however, make a rough running surface that can cause breakdowns. Multiple electrical connections cause voltage drop and contribute to more frequent electrical problems. For permanent circuits, the track sections are welded to prevent such irregularities.

- **Wood**: tracks of this material are completely handmade and therefore require a higher cost of effort and money. These circuits have not voltage problems or irregularities in the grooves. Each circuit manufactured is unique, there are not prefabricated track sections and the decoration is more original (Figure 13).
- **Mixed**: these circuits incorporate the advantages of wooden track and the versatility of plastic circuits. The main disadvantages are its high price and limited modification, but mixed tracks have the quality to be removed for better portability.

2.7.2 Modifications

There are various modifications that can be performed in slot tracks and hobbyists have great imagination to make these modifications.

Some circuits have track sections immersed in distilled water to simulate fording a river. Distilled water is not electrically conductive, but the electronic parts of the track must be protected to avoid damage.

There are also enthusiast who sprinkled flour or cocoa powder in the circuits to simulate snow or dirt in sections. The problem is that the sugar content in cocoa powder can lock the vehicle motor if is not suitably protected, due to high temperature.

It is recommended to equip vehicles with studded wheels and not mix the flour and water effects on the same circuit, because these are mixed inside the cars and on the track, hindering the proper operation.

Figure 13. *Handmade wooden rally track, inspired by the Targa Florio* [29].

2.8 Competitions classes and rules

Slot racing competitions are competitive manifestation of this hobby. Reproductions of real car models are used, although some slot cars are specifically designed for these competitions. Generally, the most famous types of slot racing competition classes are listed below:

- **Rally**: in this competition cars drive on a track as fast as possible. Rallies can be run on open tracks of indefinite length, but the technical difficulty of maintaining a large slot circuit is very high. Therefore, it is more common to run several times in small tracks.
- **Hill climbing**: in hill climb competitions the hobbyist competes against the clock to complete an uphill slot track. There are several categories and rally and speed are used in this competition. Normally each participant runs the circuit three times and the best time obtained is used for final classification.
- **Speed**: this competition is run on 8 lanes speed tracks where up to eight participants can compete simultaneously (Figure 14). The purpose of speed competitions is to give as many laps in a given time. The circuits can be extremely long and categories are divided by type of slot vehicle.

As the rules used in competitions, there are many different local, regional, national, and international organizations. Each slot association has its own rules, so just highlight the name of the principal international slot racing organizations, such as the United Slot Racers Association (USRA), the International Slot Racing Association (ISRA) and the British Slot Car Racing Association (BSCRA).

Figure 14. *8-lane speed track in Helsinki (Finland)* [30].

2.9 Hobbyists

In the world of the slot racing there are various types of fans, in some cases are genuine enthusiasts. Main characteristics of each type of hobbyist are indicated below:

- **Amateurs**: these hobbyists are inexperienced in slot racing, they do not yet have specific preferences about tracks, slot vehicles or manufacturers and invest little money in the hobby.
- **Collectors**: they are fans who collect numerous models of slot cars according to different criteria (car brands, drivers, teams or competition classes). Collectors also restore their slot vehicles and document them according to their characteristics. There are private collections with over 4000 different slot cars [11].
- **Competitors**: these hobbyists are the most competitive and they participate in many slot competitions. The possibilities of modifying a slot vehicle are numerous because there are a lot of car parts available on the market. Competitors invest money in new replacement to improve the efficiency of their vehicles and spend many hours training.
- **Modelers**: also known as designers, they create new slot car models that do not exist in the market for their own enjoyment. Modelers like the aesthetics of the slot cars rather than the mechanical part. In the world of slot, this hobby is called "scratch building".
- **Assemblers**: they are responsible for the assembly of slot circuits in competitions or manufacture track sections. Some assemblers build real circuits existing in the world of auto racing and others fans design circuits with their own imagination.

2.10 Manufacturers

In Europe, there are three major slot manufacturers: *Ninco*, *Scalextric* and *Carrera*, which offer a wide range of tracks, vehicles and accessories [18]. These brands manufactured slot products that increase the attractiveness and realism of slot car racing. But there are other manufacturers with wide recognition in manufacturing material for slot, such as:

2.10.1 Carrera

Carrera is a slot manufacturer based in Fürth, Germany which dominated the German markets in the 1960s and 1970s, due to effective marketing, also at the nearby *Nuremberg International Toy Fair*.

Currently it is characterized by the quality of its products and the wide sections of track that allows driving 1/24 scale slot vehicles.

2.10.2 Fly Slot

Fly Slot is a Spanish manufacturer famous for being the pioneer in commercializing reproductions of 1/32 scale slot cars and trucks, substantially different from the rest of the brands in the extreme quality of the products, which are appreciated by the customers and as the best in terms of number of details, technical perfection and realism to the original models.

Nowadays, there are other brands on the market that have matched the quality and faithfulness that characterized *Fly Slot*, so sales are not as numerous as before.

2.10.3 Ninco

Ninco is a Spanish manufacturer of 1/32 scale slot cars and tracks. The company was established in 1993 in Spain, by two former employees of *Exin* [14].

This manufacturer has its own criteria in analog and digital slot, but manufactures track adapters allow the compatibility of their products with other brands in analog format. Their digital slot tracks can be combined with other manufacturers, provided that the digital accessories of the track are *Ninco Digital* brand.

2.10.4 Racer

Racer is an Italian slot car manufacturer, known for its high quality fully detailed in the resin used to manufacture its slot vehicles and the efficiency of these. All cars are equipped with plastic chassis and aluminum rims.

2.10.5 Scalextric

Scalextric is a British manufacturer of analog and digital slot systems which first appeared in the late 1950s, as a creation of British firm *Minimodels Ltd*.

The brand name is a contraction of “scale x” (or variable scale) and “electric” (because it requires electricity to operate). The “Scalex” name was chosen because initially the scale of the car models was very variable. Finally, it was the 1/32 scale which acquired more popularity [13].

The name of the current owner of the brand is *Hornby* for distribution and manufacturing worldwide, except in Spain, where for historical reasons and problems of copyright Hornby products are marketed under the brand *Superslot*.

In Spain, *Educa Borras S.A.* is the company responsible for the manufacture and marketing under the name of *Scalextric*. Outside the country, this company cannot market

with the name of *Scalextric*, because *Hornby* owns the international copyrights, so *Educa Borrás S.A.* uses other brand name called *SCX*.

3. ANALYSIS OF DIGITAL SLOT CAR SYSTEMS

This chapter presents a thorough analysis of the different slot car controllers available today, starting with an overview of the analog slot car controllers made by amateurs, continuing with commercial slot car controllers that develop the main specialized manufacturers and making a research of the latest advances in digital slot car controllers. In addition, a comparison of selected slot car system is performed compared to other commercially available, detailing its advantages and disadvantages.

3.1 Introduction

Slot car systems had significant changes from the second half of the 20th century, when one of its essential components was developed, the controller, which provided better control of slot vehicles. Originally, the speed control of slot cars was "all or nothing", but these first controllers laid a solid foundation that still persists today. These primitive controllers were composed of analog electronics, such as resistors and switches, and were specially designed for analog slot competitions. Nowadays, there are primitive controllers with electronic components modified to suit the new electric ranges that use the new slot systems. With the advancement of electronics and the creation of new components, controls were developed with greater sensitivity incorporating potentiometers to change the acceleration of the vehicle in real time during the race.

In the 21st century, a revolutionary innovation highlights in slot systems, the creation of new digital slot system. This system works with a multifunctional digital control unit, which is responsible for providing "intelligence" to the system and to make all possible tasks: selection of different driving modes, sending information to the track and vehicles, time management and refueling, as well as other tasks that are explained in detail in subsequent pages.

3.2 Digital slot car systems

Important aspects of the various devices used to control slot cars are presented in this section. Firstly, slot car controllers are analyzed, differentiating their electronic devices in analog or digital. Multifunction control units of the system are also analyzed, because they are responsible to bring more realism to the hobby.

This section focuses on the digital slot system, due to its complexity and better features that make it the most innovative alternative to traditional analog slot system, as present-

ed in the previous chapter. Thus, a detailed study of the main control systems for digital slot is shown below.

3.2.1 Analog controllers

Analog controllers have undergone numerous changes over time in terms of design and structure, but its theoretical operating principle has remained almost unaltered over the years. As the technical aspects of operation were already discussed in the previous chapter, this analysis is addressed to show their main features.

Analog controller base their origin in telegraphs. The first slot vehicles were driven by rudimentary controllers made of old telegraph rheostats [31].

In 1957, commercial slot controllers were equipped with controllers with a thumb button. These controllers were “all or nothing” position, so requiring the driver to blip the throttle for intermediate speeds. Later versions also included intermediate position buttons for intermediate speed, and one late version used a buzzer mechanism to provide better speed control.

From 1959, most 1/64 scale slot cars were controlled by a miniature steering wheel was placed on a table or a manual rheostat, which gave precise control of the speed. These types of controllers allowed an adjustment more sensitive to the car's speed range, but still lacked fully regulated behavior.

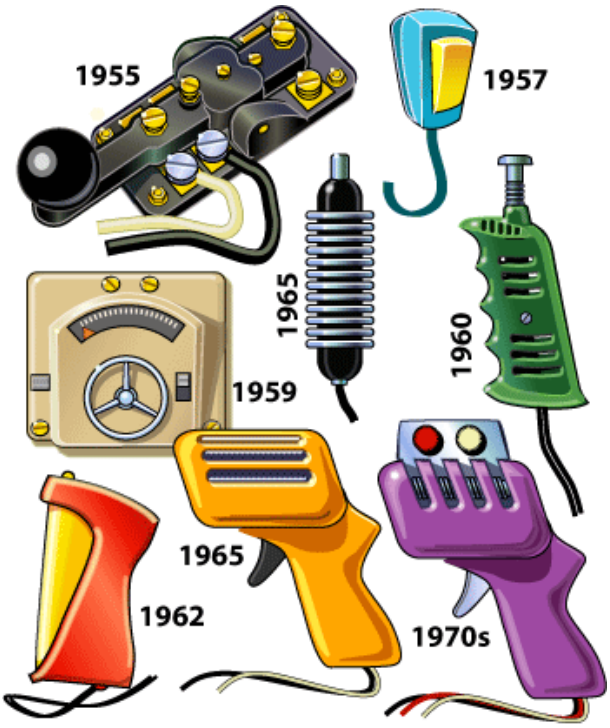

Figure 15. *Types of slot car controllers* [32].

Around 1960, controllers made with portable rheostats began to appear in slot competitions. The first prototypes were vertical and handled with the thumb finger, keeping the rheostat held by the handle. Other less common controllers were also invented as a command to be handled with full hand grip, manage the speed depending on the pressure exerted on the grip.

In 1965, the "Russkit" controller appeared and it introduced a new trigger operated manually by the finger. This controller was a landmark in the world of slot and laid the design foundations of controllers, which have survived to our times. The controller was handled pulling the trigger with the index finger and holding it with the palm of the hand through the grip, like a weapon is held. At the top was a vent for cooling the rheostat. For an adequate response, the controller should be adjusted to each individual vehicle because rheostat resistance ranges were not the same in all cases.

In the 1970s, a new type of analog controllers appeared on the market. Advances in electronics provided improvements such as the ability to modify the sensitivity of throttle or brake in slot vehicles and compatibility with most slot car systems manufactured.

After exhaustive analysis carried out in this thesis, it can be said that the trends that have followed different manufacturers of slot controllers are quite similar.

On the one hand, emphasizes the large number of slot vehicles, the realism of the tracks and their decorations, the different possibilities of improving motor vehicles getting more acceleration with miniature mechanical parts as used in real racing cars and countless accessories give this hobby more fans.

On the other hand, there are essential components for this hobby that have not improved substantially over time. As shown below, slot controllers made by principal manufacturers have few differences from those used in 1970, even some of them still keep the same design.

Scalextric

Controllers manufactured by this Spanish brand differ little in their classic versions for analog system with the latest developments, designed for digital slot, marketed under the name "Scalextric, the Digital System".

The controller design remain is handle grip and it adapts to the ergonomics of the hand. Its operation is based on a trigger which is pressed by the index finger regulating rheostat content inside the controller. This pressure serves to describe a determined acceleration curve that allows the user to control the slot vehicle speed [33].

In the upper back of the controller, there is a button that serves to change the lane through which circulate the cars and only acts on sections of the circuit enabled for it.

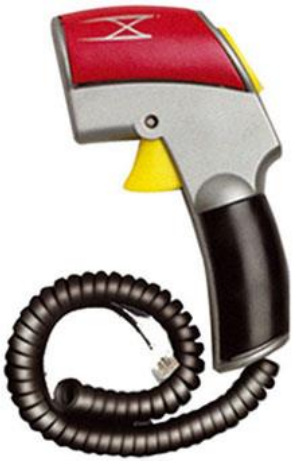

Figure 16. *Scalextric, the Digital System slot car controller* [34].

The controller communicates via 4-wire cable and a modular connector 4P4C for connecting the controller to the multi-function digital control unit.

Carrera

This German manufacturer of slot car systems stands for aesthetics and performance of its controllers, which are very similar to those made in early 1960 with vertical resistors.

Carrera controller was handled pulling the trigger with the thumb and holding it with the palm of the hand through the grip, as a handle. In this case, the index finger is responsible for press the "overtaking" button to change lanes. The controller communicates via 4-wire cable and a modified modular connector 6P6C for connecting the controller to the multi-function digital control unit.

As a novelty, *Carrera* has wireless version of their controls based on radio frequency communications at 2.4 GHz, with a maximum operating range of 15 meters, 8 hours of operation and rechargeable lithium polymer battery.

Figure 17. *Carrera Digital (left), Evolution and Wireless (right) slot car controllers* [35].

Ninco

Ninco is a Spanish manufacturer with headquarters in Barcelona. Like the previous manufacturers presented their controllers are based on the traditional functioning that identifies since the 20th century.

In the aesthetics of their controllers highlights how made the handle, which has a sleeker finish and can be customized using a removable plastic sheets of different colors.

In the classical slot controller called "N-Digital", also has a trigger for controlling speed and braking. It has a button for overtaking, which allows the car to change lanes.

This controller also communicates via 4-wire cable and a modular connector 4P4C for connecting the controller to the multifunction digital control unit.

After the proliferation of specialty slot shops and fans which modified their original controls adding a potentiometer to control "N-Digital" previously shown, the company improved its classic controller with a new version called "Progressive".

This new version has only a substantial improvement, "Progressive" controller includes a potentiometer inserted on the side that serves for adjusting engine power in real time. The potentiometer allows to adapt the pilot's needs or the type of conduction in the track. By simply turning the potentiometer, the engine power is modified with different output levels ranging from a minimum to a maximum value range.

As for its operating principle is the same as the "N-Digital" controller. It also has a trigger to regulate the speed and braking of slot vehicles, and another button to overtake. This controller also communicates via 4-wire cable and a modular connector 4P4C for connecting the controller to the multi-function digital control unit.

Figure 18. *N-Digital (left) and Progressive slot car controllers* [39].

3.2.2 Digital controllers

New advances in electronics have hardly improved the world of digital Slot, few manufacturers that have adapted to changing times improving the characteristics of its controllers. Today, the wide acceptance and fame of modern video games consoles in the entertainment industry make it difficult to compete against them.

Highlights some examples of relatively modern manufacturers have begun to innovate in the composition and operation of slot controllers. The production of these controllers is very limited and only have a reputation in the most advanced sectors of hobbyists, there are quite elaborate designs. Some are hand produced so its price is high, and also require special settings for connectivity and correct operation in the various slot tracks depending on the manufacturer.

Controllers presented below are examples of designs of digital electronic controls that today can purchased, but with difficulty:

Ghost controller

The ghost controller is also called automatic controller, because it allows a slot vehicle moves autonomously by the circuit. This automatic controller has various applications for training and development the new engines of vehicles, cleaning the rails of the tracks and allows more entertainment circuits with few drivers.

As its analog version, ghost slot car controller is manufactured by craft hobbyist and slot shops using traditional methods, so it is not approved by the official brands of the different slot car systems.

This controller is made of resistors, capacitors and potentiometers for speed control. It also has a relay for the lane change and an integrated circuit 555, which allows the timing of lane change. It is powered by a 9 V battery and connection is via 4-wire cable and a modular connector 4P4C.

Figure 19. *Automatic or ghost slot car controller*[1].

SCP-2 controller

The "SCP-2" controller is manufactured by the Italian manufacturer *Slot.it* and it is a digital controller for slot vehicles. Its operating principle is based on a control system that uses pulse width modulation (PWM) for the acceleration and braking of the vehicle.

This controller has no actual electrical contact, due to the trigger position is read magnetically using a linear Hall Effect sensor. Most slot systems are compatible with "SCP-2" controller because it works through a system of interchangeable cartridges containing appropriate electronic circuitry for each brand. The digital cartridge is universal for major commercial slot digital systems, only changing the connectors.

The controller offers different driving modes, to have a custom control of the slot vehicle:

- **Linear**: the relationship between the cursor and the output voltage is an ascending line. The controller, when the trigger is fully depressed, delivers maximum power on the track.
- **Linear with speed limiter**: the relationship between the cursor and the output voltage is an ascending line, but when the trigger is fully depressed, optionally, the voltage can be reduced up to 60% of available voltage.
- **Curve**: it is a sophisticated driving mode with total control in the response curve. The relationship between the trigger and the output voltage is not a straight line but may be concave or convex.
- **Ghost car**: this mode allows automatic speed control, it is useful for driving cars on the circuit automatically or to test slot car engines.

Slot.it controller can be used with major commercial slot digital systems, which between them are incompatible. Besides connection to the multifunction digital control unit, this controller is required external power source for its operation.

Figure 20. *SCP-2 slot car controller, manufactured by Slot.it* [36].

Scorpius Wireless controller

Defining “Scorpius Wireless” as slot controller is not entirely correct, because it is a complete control system for digital slot, manufactured by the company *Australian Slotcar Technologies*.

This system consists of four different components, specifically designed for installation in wood or to adapt plastic tracks of the leading manufacturers (*Ninco*, *Scalextric* and *Carrera*). Scorpius Wireless system is designed to work with a computer, because it has its own programming software for the controller. The software allows run with up to 24 cars at a time, according to the manufacturer's specifications [37].

The complete slot car system is composed of:

- **Controller**: the trigger is operated by a Hall Effect sensor and the controller is connected by radio frequency 2.4 GHz, it has LCD display and vibration alert system.
- **Chip**: is responsible for providing wireless radio frequency 2.4 GHz.
- **Lane Brain**: is responsible for monitoring the lane change, with wireless connection. The user should take care of the installation of Lane Brain device in each circuit.
- **Dongle**: it is a USB antenna that communicates wirelessly and allows the exchange of information between different components of “Scorpius Wireless” system.

As for the disadvantages of “Scorpius Wireless” system should be emphasized that it is not an official system of any trademark, so are the fans themselves who have to install it on their circuits of other brands or made by themselves, modifying the original tracks for proper operation. It also has the disadvantage of its limited production and the high price of the whole slot car system.

Figure 21. *Scorpius Wireless controller* [37].

3.3 Slot car system selected

In this section, the control system that focuses on this thesis is discussed, highlighting the main advantages and disadvantages that identify the system. The different parts forming the slot car system selected are also explained, as the circuits, the control unit and other accessories.

3.3.1 Advantages and disadvantages

After analyzing various specialized written sources such as magazines and books as well as digital, whether forums, blogs or online journals, it is concluded that the brand selected for the project has great prestige among the Spanish hobbyist and increasingly among international slot fans.

Surveys conducted by specialized sources in the field of digital slot as the premier web-site dedicated to digital slot in Spain, assigned to the digital system *Ninco* a rating of notable above all its competitors [38].

There are several reasons why *Ninco* is the preferred brand of slot car system in Spain, including the following:

- Better connection between tracks, with lower voltage losses than its main rival manufacturers.
- The production is located in Spain, while the Spanish subsidiary of *Scalextric* focuses all its manufacturing in China.
- It has great quality finishes their cars and circuits, which overcomes the other slot companies.
- *Ninco* is a brand that takes into consideration the feedback from fans and try to solve some of the historical problems of slot systems, such as loss of tension between sections of tracks.
- There are connections and track sections that allows compatibility with other rival brands such as *Scalextric*.

In contrast, *Ninco* also has some disadvantages:

- The price of the products is higher than its main competitor, because production in Spain is more expensive than in China.
- *Ninco* not manufacture wireless slot controllers, while other brands like the German Carrera have innovated in this field.

3.3.2 Ninco digital system

In this section, the main components of *Ninco* digital system are analyzed. First, the characteristics of the different tracks are listed and then several additional accessories are presented [39]. Finally, the component that gives "intelligence" to the *Ninco* digital system, the multifunction control unit, is described.

Tracks

- **Digital Mastertrack**: this track is identical to analog version, but also includes the multifunction digital control unit, different sections of track, 4 lane changes track sections, 3 decoder chips, 3 controllers and a transformer to power the slot system. It can be mounted on a board that measures 2.44 x 1.22m and it has 12.8m of itinerary.

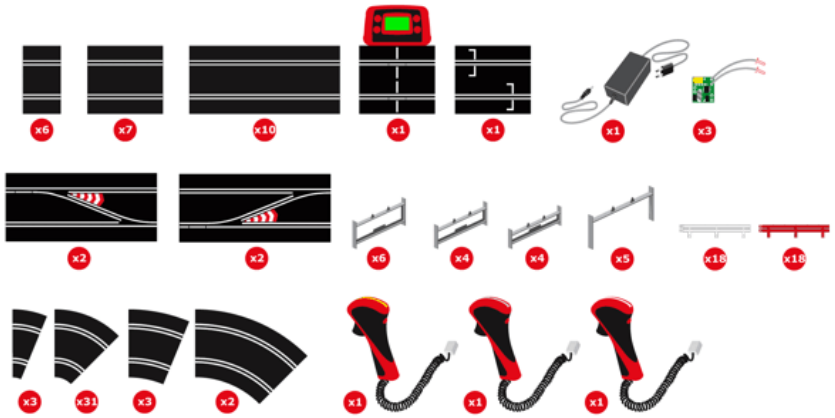

Figure 22. *Digital Mastertrack components* [40].

- **Digital Starter Box**: this track includes 2 controllers, 2 lane changes track sections and 2 vehicles digitized. Digital Starter Box is the slot system used in the process of research and tests conducted in this thesis.

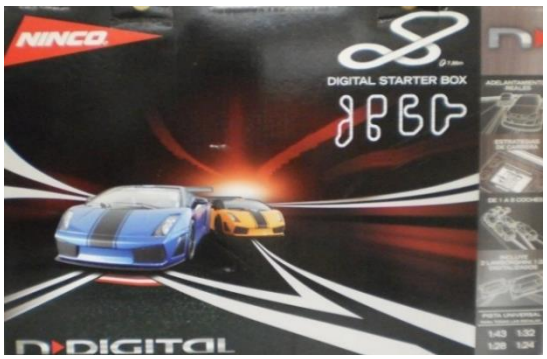

Figure 23. *Digital Starter Box*[1].

- **Kit Digital**: this kit allows to convert analog slot cars and tracks to digital. The set is formed by the multifunction control unit, 2 lane changes track sections, 3 decoder chips, 3 controllers and a transformer to power the slot system. Convert an analog slot track to digital is simple and can also be used analogue *Scalextric* section tracks with their corresponding adapters, but sometimes mix these sections can cause conductivity problems. To convert a digital slot track to analog is necessary to remove the multifunction digital control unit and install a track with analog connections. Digital slot controllers do not serve to analog slot and vice versa. Digital transformers are valid for analog slot system and analog decorative accessories are fully compatible to the digital system. In digital slot tracks, vehicles can only circulate in a predetermined direction, while in the analog are interchangeable.

Accessories

- **Controllers**: in a circuit can be connected up to 8 controllers at the same time with a single transformer, but *Ninco* recommends having 2 transformers when more than 4 vehicles are both on the same track. Controllers have a special button that allows the vehicle to change lanes or enter in the pit lane. To change lanes, it is necessary that this button is pressed when the car passes the track section intended for this.
- **Control Tower**: this tower displays the race position of each car and either the number of laps or minutes remaining in the race. Control Tower is only sold separately and it connects to the track with a special half straight track. The display is updated every time a car crosses the start/finish line, so the racing results are always displayed in real time. More than one tower can be used on the same track. It swivels 360°, so it can be positioned anywhere and still viewed easily.

Figure 24. *Control Tower* [39].

- **Pit Lane**: it adds another element to the slot car racing requiring each car to perform a certain number of pit stops on the track. The only limitation of the pit

lane is that only one car can pit at a time. If two cars are in the pit lane at the same time, the second car in will immediately start refueling when the first car is finished refueling which is similar to real race where cars can get delayed either entering or exiting the pit stop with other cars in the pit stop at the same time. The operation is better when racing with a lot of drivers, to pit either a few laps early or to short pit the first stop to minimize the possibility of needing to pit when the pit lane is being used already.

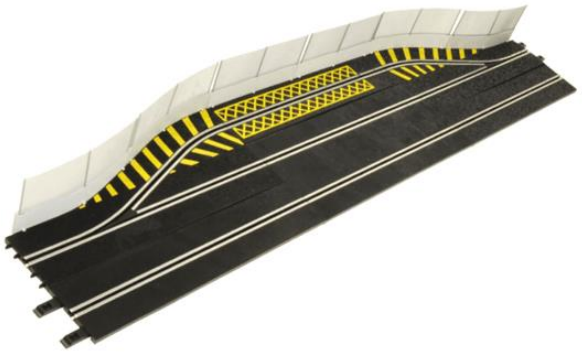

Figure 25. *Pit Lane* [39].

- **Lane Change**: *Ninco* Lane Change track sections are different from other slot manufacturers because this track section is responsible for moving a flap that allows the vehicle to change lanes, while in other slot systems the vehicle guide flag is the responsible. There are three different lane changes track sections. The most famous is the Double Lane Change, although there is also Simple Lane Change. The latest innovation is the Double Lane Change Curve.

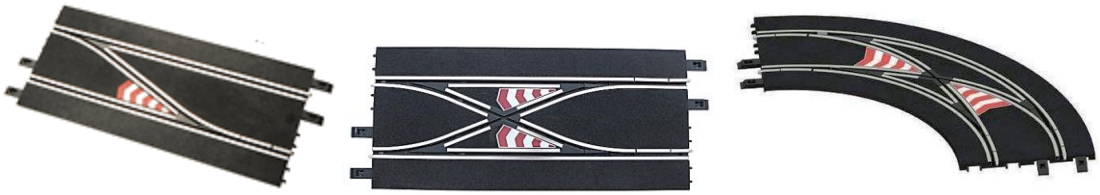

Figure 26. *Simple (left), Double and Double Curve (right) Lane Changes* [39].

Its operating principle is simple and to make a lane change just press the button on the back of the controller before reaching change and holding it down until the change is done completely.

- **Decoder chip**: it is simple to place on slot vehicles that have space for this specific purpose. To work correctly, only have to connect 2 wires to the guide and 2 other wires to the motor. Also incorporates two additional contacts where optional lighting can be connected in the event that the vehicle is equipped with this feature.

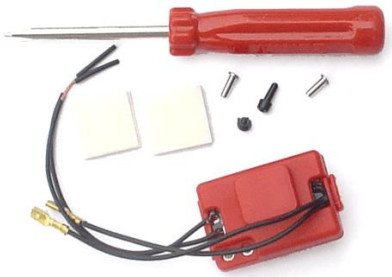

Figure 27. *Decoder chip and equipment* [39].

- **Multi-Lane Sensor**: this track section is for 4-lane tracks and it allows the multifunction digital control unit to keep track of all the racing action on lanes 3 and 4. Multi-Lane Sensor is necessary to have a finish line with four lanes. It has electrical reliability and it is very easy to assemble and disassemble.

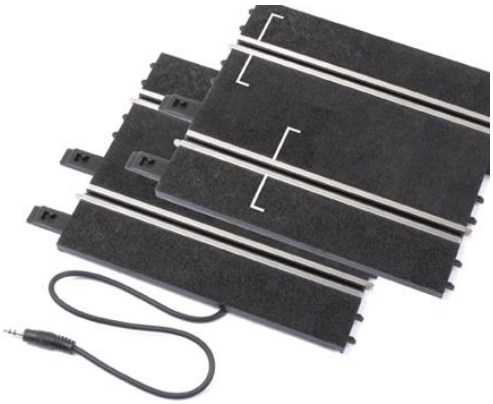

Figure 28. *Multi-Lane Sensor* [39].

Multifunction Digital Control Unit

Multifunction Digital Control Unit is the heart of the slot system, allowing race control for up to 8 lanes of racing. The multifunction control unit display manages 7 different types of races for individual and multi-players. It provides racing information and sound signals for all drivers: time, lap and fastest lap, fuel consumption simulation, management for cars with and without lights, fastest reaction speed from driver to car. The transformer provides the power for up to four 4 cars [40].

The different modes of racing competition that gives the control unit are listed below:

- **Grand Prix (GP)**: in this race mode can select the race distance in laps or total time that you select. Each driver's info including top speed, lap speed, fastest lap, and position can be shown.

- **World Rally (WR)**: it is a single rally point to point stage with individual times for up to 8 racers. Display give finishing positions and individual information for each driver.
- **Fast Lap (FL)**: this is the race mode for selecting qualifying times for race positions. Used for qualifying-style competitions, personal info for each driver is shown.
- **Non Stop (NS)**: it is an endurance racing. It can run indefinitely and continually maintains driver info for fastest lap and race positions.
- **Pit Strategy (PS)**: this mode is just like the Gran Prix mode, but like real racing. It necessary have Pit Lane installed for this mode to work because the slot car have to enter in the pit lane to refuel.
- **Arcade Group (AG)**: it is a staged event for up to 8 drivers that will knock out the slowest car after each stage, leaving the final two racers for the main event.
- **Arcade Single (AS)**: this race mode allows program the time or laps and race against the multifunction control unit. At the end of each race, the system resets itself with the lower time.

Figure 29. *Ninco Multifunction Digital Control Unit*[1].

4. RESEARCH METHODOLOGY AND CONTROLLER FEATURES

4.1 Introduction

This chapter details the methodology used in the research process of the Master's Thesis and presents the different basic characteristics of the new controller designed for digital slot car racing.

First, the chapter presents the research methodology used to produce the controller, indicating the stages performed for the analysis, design and implementation. In this section, the research methodology conducted to understand the principles of operation of digital slot system studied is also presented. The most significant laboratory tests and preliminary prototypes made to test hypotheses are shown below.

Finally, an overview of the features of the slot controller designed and its advantages over other commercial slot controllers previously discussed in the chapter 3 are shown here.

4.2 Research methodology

In the next section, the steps followed to design and implement the digital slot controller made in the project are listed. The controller was designed following a series of stages, starting with the requirements specification and ending with the product implementation.

Then, the different activities that are part of the methodology to consider in designing prototypes are listed:

- **Requirements specification**: in this phase the minimum requirements that should cover the project to develop it and their general characteristics are indicated. An overview of the project is shown and it deepens in later phases. Parameters such as basic functions and features of the visual interface are also specified.
- **Software and hardware architecture design**: it is very important the software design for the correct operation of the processing system that gives intelligence to the controller and its hardware architecture. This phase is theoretical and requires constant debugging and feedback for the improvement of it throughout the research process.

- **Software and hardware development**: once submitted the basic features of the controller it is necessary to continue with the development phase. In this phase the program code is completed and the electronic components are selected to implement the project. The printed circuit board, which contains the various electronic circuits and other necessary components are also designed here.
- **Testing and debugging**: after completing the development of control hardware and software, different simulations and tests of the functional blocks that make up the slot controller are made. Thus errors that may affect the operation of the assembly are avoided. After assembly of the controller, experiments were conducted on a slot track to check the correct functioning of the controlled vehicle, prior to its final implementation.
- **Design and implementation of the printed circuit board (PCB)**: another important phase is the design and manufacture of printed circuit board, once the correct operation of the implemented circuit is checked on a breadboard. After printing the PCB, the electrical conductivity of the copper tracks is tested and then different electronic components are soldered. Finally, a last check of the finished slot controller is performed to verify proper operation.
- **Experiments and final tests**: when the digital slot controller is fully assembled, the final experiments are conducted to verify proper operation of the controller and to check that its performance is as in previous phases.

4.3 Research process

Today the world of slot car racing still has quite a reputation among hobbyists and collectors. For this reason, the technical details of accessories and systems are kept secretly, to avoid the risk of plagiarism by different slot manufactures which sometimes are more worried about the innovations of other brands, instead of focusing on their own development.

Therefore, for the realization of this thesis it was not possible to base research on a variety of reliable sources about slot controllers. The information on the operating principles of controllers has been obtained by the exhaustive analysis of commercial slot controllers and numerous tests in the laboratory.

In this section, the elaborate process of research conducted to prepare the slot controller addressed in this project and their bases of operation are described.

4.3.1 Test slot track

For the various tests and experiments carried out along the research process of this project, a complete system of digital slot car manufactured by the Spanish company *Ninco* was used.

Figure 30. *Slot track used in laboratory tests*[1].

The central laboratory where the tests were performed is located at the Technical University of Cartagena (UPCT), in Spain, and it has different material for the realization of measures and manufacturing of the slot controller.

4.3.2 Connection

As a first point, a study of the type of connection used by slot controllers to the multifunction digital control unit of the system was performed. The controller used is manufactured by *Ninco*, and for connection it uses a 4P4C connector (4 Position, 4 Conductor) which has four pins and four cables. These connectors are widely used internationally and there are different types, depending on the number of pins and wires that make it up.

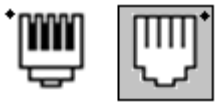

Figure 31. *4P4C connectors*[2].

4P4C connectors are also known as Registered Jack RJ10 or RJ22 and are commonly used in telephone handsets. The slot controller uses a 4P4C male connector, while the multifunction digital control unit uses a 4P4C female connector.

4.3.3 Basic slot controller diagram

The next step was to analyze the basic electronic components that composed the commercial controller and thus examining if the hypothesis of the operating principle of the slot controller was correct.

The schematic of Figure 33 shows the basic electronic components of a digital slot controller manufactured by *Ninco*. This controller has a set of resistors in series, a conductor cursor, a special button to change lanes, a small vibrator motor and a filter capacitor.

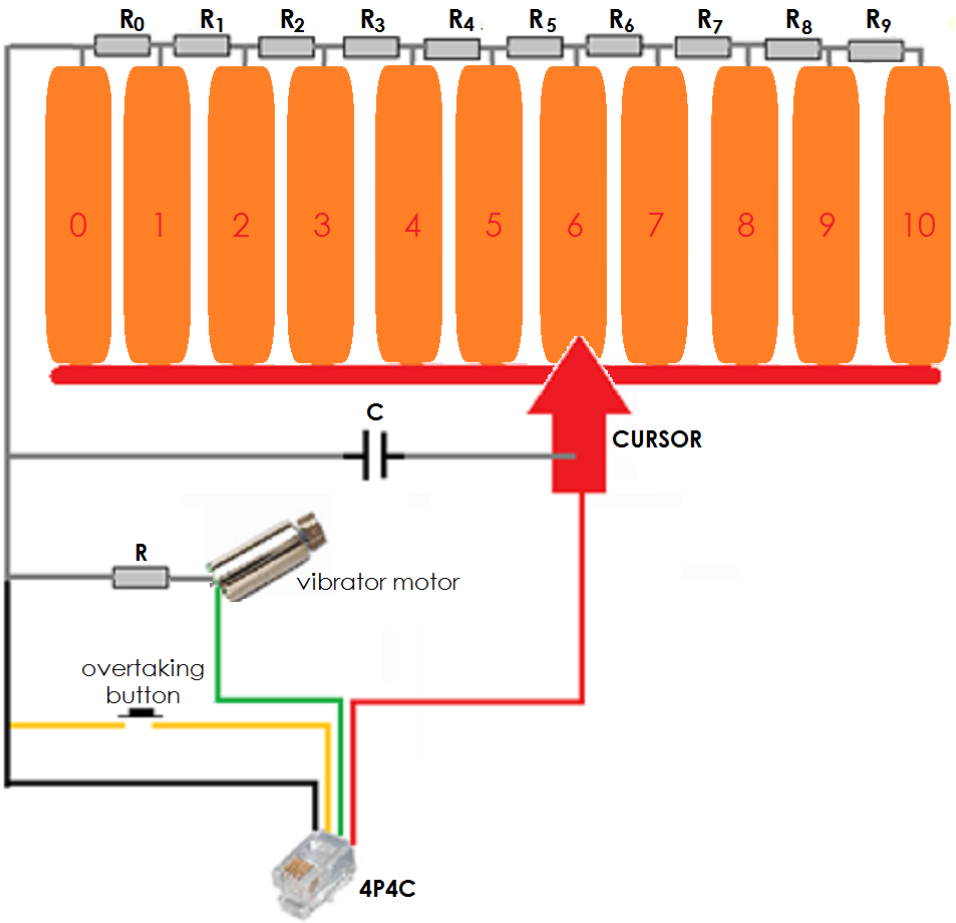

Figure 32. *Basic diagram of Ninco slot controller*[2].

The alphanumeric and color code used for referencing the wires that connect the controller to the multifunction control unit is shown below:

- **Connector pin 1 (CN1)**: black wire. It is responsible for supplying 5.05V to the slot controller input from the multifunction control unit.
- **Connector pin 2 (CN2)**: red wire. This wire provides the output voltage which is applied to the slot vehicle motor.
- **Connector pin 3 (CN3)**: yellow wire. It is the output that activates the lane change.
- **Connector pin 4 (CN4)**: green wire. This wire is connected to a small vibrator motor inside the controller.

As shown in the diagram, the slot controller is composed of 10 resistors connected in series and a slider which performs the function of voltage divider, regulating the output voltage with an accuracy of 10 steps.

The resistance values used in the controller are:
R = 220 Ω.
R_0 = 470 Ω.

$\mathbf{R_1} = \mathbf{R_2} = 470\ \Omega$.
$\mathbf{R_3} = \mathbf{R_4} = 680\ \Omega$.
$\mathbf{R_5} = \mathbf{R_6} = 1\ k\Omega$.
$\mathbf{R_7} = 1'5\ k\Omega$.
$\mathbf{R_8} = 2'2\ k\Omega$.
$\mathbf{R_9} = 3'3\ k\Omega$.

Actuating the trigger of the controller, the cursor is slid on the surface of copper plate extension that serves variable resistance from R0 to R9 and it gives a certain output voltage by the wire CN2. If the trigger is pressed to the end, the output voltage generated allows the higher speed reached by the vehicle slot.

In the black conductor (CN1), there is always 5.05V applied by the multifunction control unit. After doing the necessary electrical measurements, it is concluded that the voltage applied to the motor of the slot vehicle is the potential difference obtained between the wires CN1 and CN2, varying as follows:

- **Zero speed**: without pressing the trigger, the output voltage is 5.05V measuring CN2 connected to the ground of the multifunction control unit or 0V measuring the electric potential difference between the wires CN1 and CN2.
- **Maximum speed**: with the trigger pressed, the output voltage is 1.35V measuring CN2 connected to the ground of the multifunction control unit or 3.7V measuring the electrical potential difference between the wires CN1 and CN2.

The vibrator motor located in the slot controller operates only when the multifunction control unit sends a signal, in the case of fast lap and final lap. This signal activates the ground of the vibrator motor, causing it to turn on, because the motor is always supplied with 5V.

There is also a button to activate the lane change. When it is pressed, the circuit is closed and passes a 5V signal arriving at the multifunction control unit by the wire CN3. The multifunction digital control unit processes the received signal and activates the lane changes on the track if there is a slot car on these track sections.

Laboratory tests were also performed with an advanced version of *Ninco* controller equipped with linear potentiometer called "Progressive", previously presented in chapter 3. In this controller, the potentiometer is in series with resistor R_0 indicated above and provides the ability to modify the acceleration curve.

4.3.4 Preview prototype

Several experimental prototypes were made to test it practically on a slot track and thus the hypotheses formulated theoretically, based on original research reported in previous projects [38].

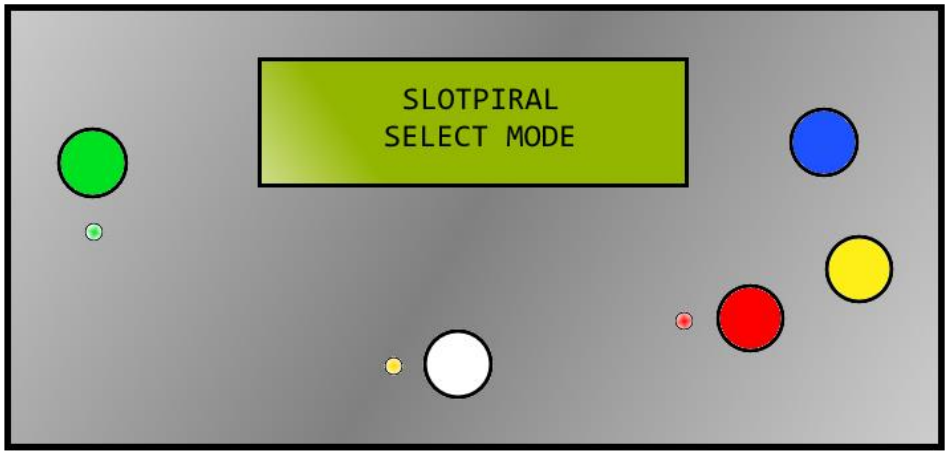

Figure 33. *Slotpiral front panel*[2].

The prototype controller for digital slot that was part of the origin of this Master's Thesis was also implemented. This prototype was called "Slotpiral" (Figure 33) and its main basic features are listed below due to its importance to the present project.

The front panel of the "Slotpiral" controller consists of various electronic components such as resistors, push buttons and LEDs. It also has a LCD screen for easy operation. The different requirements of "Slotpiral" are listed below:

Push buttons

The front panel of the controller was implemented using round push buttons, which is an innovation over the triggers used by the previous slot controllers. The structure of the controller front panel was composed of the following buttons:

- **Throttle**: it is the red push button and it performs the function of accelerating the slot car speed gradually. While the throttle button is pressed the vehicle speed increases to the maximum speed.
- **Brake**: it is the yellow push button and it performs the function of braking the slot vehicle. While the brake push button is pressed the car speed decreases rapidly to stop the car.
- **Turbo**: it is the blue push button and it performs the function of accelerating the slot car to the maximum speed. When the turbo button is pressed, the vehicle speed increases to maximum instantly.
- **Pass**: it is the green push button and it performs the function of passing other cars. When the pass button is pressed a signal to the multifunction control unit is sent to activate the lane change section in the circuit.
- **Mode**: it is the white push button and it performs the function of selecting different driving modes that the controller allows.

LED lighting

As shown in Figure 33, three light-emitting diodes are also necessary to identify the different actions of the slot controller. The acceleration is marked with a red LED and change lanes with a green LED. The different driving modes of the controller are characterized with different blinking of an orange LED.

LCD screen

“Slotpiral” includes a liquid crystal display for easier operation and selection of driving modes of the controller. The display shows the name of each driving mode available and a simple graphic speedometer.

Control interface

The control interface of this slot controller is composed of a hardware and software open source platform called Arduino Uno. This platform is a microcontroller board based on the ATmega328 and it has input and output pins specified in more detail in the precursor project of this Master’s Thesis [38]. The software interface development of “Slotpiral” was discussed in another project that is also recommended [41].

Battery system

This controller is powered by two 9V batteries.

Communication

“Slotpiral” controller uses to communicate a cable consists of four copper wires and a 4P4C connector for connecting the slot controller with the multifunction digital control unit.

4.3.5 Final slot car controller

After listing the basic features of the prototype controller “Slotpiral” in the previous section, now features for the ultimate controller that is designed for this thesis are presented.

The name of the final version of the slot controller is “Slotpiral R-evolution” and it has many new features over the previous prototype, highlighting its wireless communication system and its new TFT LCD color screen.The front panel of “Slotpiral R-evolution” controller has a similar design to the previous prototype, due to its simple interface and usability.

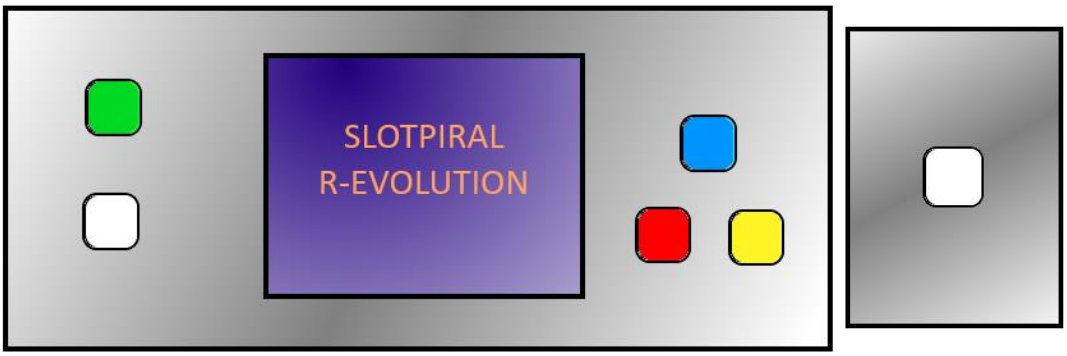

Figure 34. *Slotpiral R-evolution transmitter (left) and receiver*[2].

The main features of the final slot controller are listed below and they will be explained in detail in the following chapters.

Push buttons

The front panel of the "Slotpiral R-evolution" controller was implemented using square push buttons. The structure of the controller front panel was composed of the following buttons in the transmitter module:

- **Throttle**: it is the red push button and it performs the function of accelerating the slot car speed gradually. While the throttle button is pressed the vehicle speed increases to the maximum speed with an acceleration according to the driving mode selected.
- **Brake**: it is the yellow push button and it performs the function of braking the slot vehicle. While the brake push button is pressed the car speed decreases rapidly to stop the car. The brake button has priority over the other buttons as it is the most important, therefore when it is pressed while the other buttons, its action overrides the other.
- **Turbo**: it is the blue push button and it performs the function of accelerating the slot car to the maximum speed. When the turbo button is pressed, the vehicle speed increases to maximum instantly. The turbo button has priority over the throttle button.
- **Pass**: it is the green push button and it performs the function of passing other cars. The pass button has priority over the throttle and turbo buttons.
- **Mode**: it is the white push button and it performs the function of selecting different driving modes that the controller allows.

The receiver module has on its front panel the following push button:

- **Automatic**: it is the white push button and it performs the function to activate and deactivate the automatic mode or ghost mode, which is able to control a car without driver automatically.

TFT LCD screen

"Slotpiral R-evolution" includes a thin-film-transistor liquid-crystal display for easier operation, it is able to select the different driving modes of the controller and it displays the brake, turbo and pass actions in real time. The display also shows a graph of velocity versus time, which is updated in real time and also serves as a speedometer.

Control interface

The control interface of this "Slotpiral R-evolution" controller is also composed of a hardware and software open source platform Arduino for the transmitter module and the receiver. The Arduino board selected and the programming code used for the graphical interface of the TFT LCD display and the control interface are presented in more detail in subsequent chapters.

Battery system

The transmitter module is powered by 9V battery, to have more autonomy without physical limitations. The receiver module is supplied to the multifunction control unit via cable while the control unit is powered by a 14V transformer at the same time.

Communication

Communication system that implements "Slotpiral R-evolution" controller consists of two modules:

- **Receiver module**: it is connected to the multifunction control unit by physical means, a cable consists of four copper wires and a 4P4C connector for connecting the slot controller with the multifunction digital control unit. This module is responsible for receiving the radiofrequency signal and converts this into voltage to power the motor of the slot vehicle. The receiver has a button to activate and deactivate the automatic mode.
- **Transmitter module**: it is the front panel of the slot controller and its function is to send serial data encoded numerically for a high-speed communication in real time, without delay.

4.3.6 Improvements

As discussed in previous chapters, the evolution of analog and digital slot controllers is characterized by a state of the art unchanged in recent decades. There are few innovations in slot controllers, simply focus on slightly modify its appearance.

The controller "Slotpiral R-evolution" performs a variety of hardware and software innovations, without any previous reference among which are the following:

- “Slotpiral R-evolution” uses push buttons to control the slot vehicle instead of the classic trigger. The push buttons including new actions for driving like turbo and brake push buttons, which had never been implemented in a slot controller.
- The front panel has an intuitive interface that facilitates the management of slot cars. The push buttons are placed prioritizing the most used in races on the right side and secondary buttons on the left side.
- The controller has multiple driving modes for modifying the parameters of the car motor and to configure them for each type of slot track.
- Graphical interface TFT LCD screen, which displays detailed information about the driving mode, main functions, a graph of velocity versus time and speedometer in real time.
- The slot controller communication is wireless and it is based on radio frequency technology that allows greater range than other controllers available on the market without interference.

5. HARDWARE ARCHITECTURE

This section describes the hardware architecture of the "Slotpiral R-evolution" controller and the principal elements of the processing system such as digital - analog conversion, graphical interface, front panel, power and communication system used.

Subsequently, a detailed theoretical study of the features presented in the previous chapter having regard to principles, characteristics and types available for project implementation is done. First, an overview of possible devices that can be used is presented to finish with the definitive devices finally selected.

5.1 Description of the hardware architecture

This section describes the multiple blocks that compose the hardware architecture implemented in the project for control of a digital slot vehicle.

The following figure summarizes the main blocks in which the slot controller is divided. The main blocks are the transmitter module and receiver module.

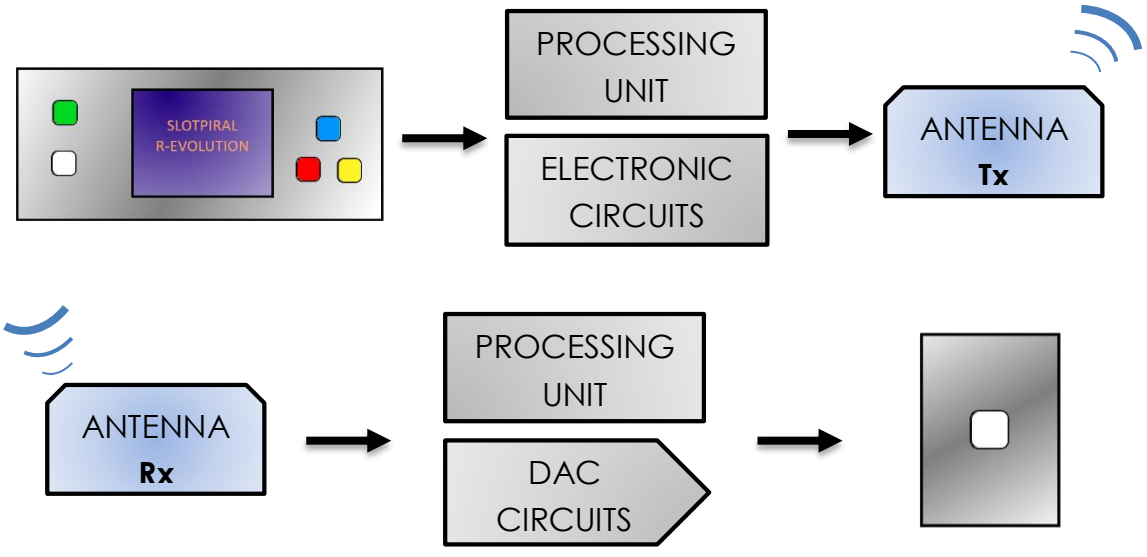

Figure 35. *Transmitter (above) and receiver hardware architecture block diagram.*

5.1.1 Transmitter module

This module consists of a front panel, a processing unit, electronic circuits and antenna (Tx) responsible for data transmission. The front panel is a control interface composed of electronic circuits, push buttons and LCD TFT screen allowing the management of the slot controller.

The processing unit processes the information of the operations on the front panel using the Arduino platform. Arduino is a microcontroller that contains the controller software. This software is responsible for controlling slot vehicles, the operation of the screen and the wireless communication. For wireless signal transmission, the radio frequency module sends serial data to the receiver module.

5.1.2 Receiver module

This module consists of a front panel, a processing unit, electronic circuits and antenna (Rx) responsible for data reception.

The antenna (Rx) is responsible for receiving the signal sent by the transmitter. After receiving the signal, the processing unit analyzes the signal by using the Arduino microcontroller platform, which provides the software to adapt it. Electronic circuits convert the received digital signal into analog using a digital-to-analog converter (DAC), with a resolution of 8 bits.

The converter is based on a ladder resistors or R-2R network that transforms the digital signal into analog. Subsequently, another circuit formed by a relay is responsible for selecting the output signal that is sent to the control unit. The final output signal is sent to the multifunction digital control unit via a 4-wire cable and a 4P4C connector, and finally the multifunction control unit sends it to a specific slot vehicle.

5.2 Processing unit

5.2.1 Microcontroller

A microcontroller is a programmable integrated circuit capable of executing commands stored in its memory. It consists of three main functional units of a computer: central processor core, memory and input / output peripherals.

There are four-bit words microcontrollers that operate at clock rate frequencies as low as 4 kHz, for power consumption in μW. Some microcontrollers may retain the functionality while waiting for an interrupt with energy consumption in “sleep” mode that reduced power consumption to nW, and it makes them suitable for applications with long battery life.

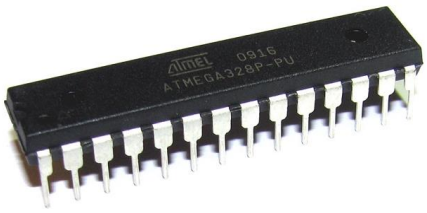

Figure 36. *Atmel microcontroller encapsulated in a 28-pin package*[1].

Microcontrollers are designed to reduce the cost and power consumption of a particular embedded system. Therefore, the size of the central processing unit, the memory and peripherals including depend on the application. They are used in automatically controlled devices or embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications, which use a separate microprocessors, memory, and input / output devices.

5.2.2 Selection of the microcontroller

To select the most suitable microcontroller used in the "Slotpiral R-evolution" slot controller, the following factors were considered:

- **Data processing**: it is required that the microcontroller can perform many critical calculations in the shortest possible time. The microcontroller chosen has to be fast enough to perform the required operations on the slot controller without delays. The central processing unit (CPU) needed for this project will use an 8-bit processor.
- **Inputs / outputs**: the microcontroller chosen must have enough inputs / outputs for implementing the project completely. To increase the number of inputs / outputs can be multiplexed, but the most effective is to choose a higher microcontroller.
- **Memory**: memory requirements of the application are divided into volatile memory (RAM), non-volatile memory (ROM or EPROM) and modifiable non-volatile memory (EEPROM). EEPROM memory can be useful to include specific information on the application, such as a serial number or different calibration parameters. To estimate the required memory is essential to program a preliminary version of the application.
- **Power consumption**: the slot controller is powered by batteries, so the power consumption of the microcontroller has to be as low as possible for efficiency.
- **Circuit board design**: the packaging of the microcontroller is important because it affects the design of the circuit board. Sometimes using an economic microcontroller can be counterproductive, because it may require additional electronic components and be more expensive.

5.2.3 Arduino platform

After analyzing the different microcontrollers on the market and using the above selection criteria, the final decision to use the Arduino platform to develop the project was taken.

Arduino is an open-source software and hardware platform code, implemented on a printed circuit board with a microcontroller and integrated development environment (IDE) designed to facilitate the use of electronics in multidisciplinary projects.

The hardware consists of an Atmel 8-bit AVR microcontroller with complementary components that facilitate programming and incorporation into other circuits. The software consists of a cross-platform development environment written in Java that derives from the Processing programming language and the Wiring projects. Arduino programs are written in C or C++ program languages and the platform is pre-programmed with a bootloader that simplifies uploading of programs to the on-chip flash memory.

Arduino can be used to develop interactive objects, taking inputs from a variety of sensors and physical outputs. The projects can be stand-alone or they can communicate with software running on a computer. It received an Honorary Mention in the Digital Communities section of the 2006 Ars Electronica Prix. The Arduino founders are: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino and David Mellis.

There are many other microcontrollers and platforms numerous advantages and applications, but it cannot compete with Arduino platform, because Arduino simplifies the process of working with microcontrollers.

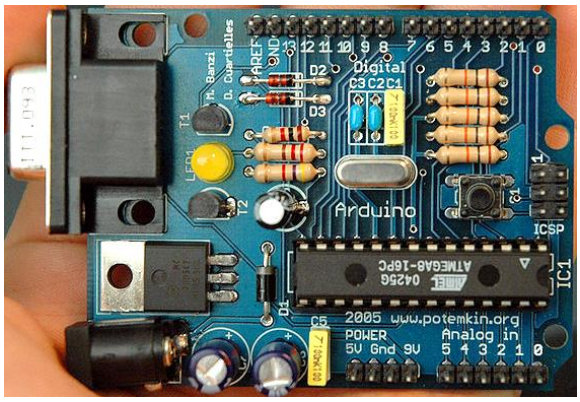

Figure 37. *Arduino Serial board* [42].

5.2.4 Arduino advantages

Arduino is a low-cost platform that has great versatility and facilitates the use of microcontrollers. Listed below are the main advantages to highlight of this software and hardware platform:

- Arduino platform has a clear programming environment but it is completed enough for advanced users, allowing it to be used to design complex electronic projects.
- Arduino boards are relatively inexpensive compared with other microcontroller platforms. This platform can be assembled by hand and schematics are available on the official website [43].
- It is a cross-platform environment, compatible with any computer that includes a USB port communication or alternatively a serial port. Arduino software runs on all major existing operating systems: Windows, Macintosh and Linux. Unlike most environments for microcontrollers which are limited to Windows.
- The Arduino software is extensible and it is published as open source, available for extension by experienced programmers, in AVR C programming language on which it is based. The language can be expanded through C++ libraries.
- The Arduino hardware is also extensible and open source. The plans for the platform are published under a Creative Commons license, so enhanced versions can be made.
- There is a lot of information to learn more about this platform available to users, such as tutorials and open-source publications available for download online. Arduino has a large community of supporters who continually generate new projects.

In summary, the advantages of the Arduino platform are many, highlighting a powerful and inexpensive hardware, an integrated development environment based on Processing and similar to C or C ++, which is unusual in 8-bit microcontrollers.

5.2.5 Arduino Nano

After analyzing the different models of the Arduino platform currently available [44], a comparative tabulation of the most important characteristics was performed and analyzed to show that Arduino version might be more convenient for its use in the slot controller:

In the table below, only the specifications of the Arduino platform most used are detailed, other platforms are not included because they are not valid for the project. All the Arduino boards presented in the Table 2 have sufficient analog and digital input / output pins to implement the slot controller designed.

Table 2. *Arduino platforms basic specifications.*

Arduino	Nano	Leonardo	Uno	Due
Digital I/O Pins	14	20	14	54
Analog Input Pins	8	12	6	12
Microcontroller	ATmega328	ATmega32u4	ATmega328	AT91SAM3X8E
Flash Memory (kB)	32	32	32	512
Clock Speed (MHz)	16	16	16	84
Dimensions (mm)	45 x 18	68,6 x 53,3	68,6 x 53,4	101,52 x 53,3
Price (€)	30	22	26	40

Microcontrollers of the different platforms are based on Atmel AVR 8-bit RISC-based microcontroller, so they have identical flash memory and clock speed. Arduino Due board is a higher level, because it has a microcontroller that exceeds the needs of this project.

The dimensions of the boards are also a factor to consider, because the size of the controller should be as comfortable as possible, so that the selected Arduino platform should not be too large. Another important factor is the price of the selected Arduino platform, considering commercial aspects it is preferable that the cost were as low as possible, conforming to the technical specifications in order to create a profitable and efficient product.

For the above reasons, the platform selected to implement the processing unit of the controller "Slotpiral R-evolution" was Arduino, because it fits perfectly with all the technical specifications for the project.

Arduino Nano specifications

The Arduino Nano is a complete circuit board based on the microcontroller Atmel ATmega328. It has more or less the same functionality of the Arduino Uno, but in a smaller package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one. The Arduino Nano was designed and is being produced by *Gravitech*.

Figure 38. *Arduino Nano board front (left) and rear* [53].

The main technical specifications of Arduino are summarized in the following table:

Table 3. *Arduino Nano technical specifications.*

Arduino Nano technical specifications	
Microcontroller	Atmel ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7 - 12V
Input Voltage (limits)	6 - 20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40mA
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2kB
EEPROM	1kB
Clock Speed	16MHz
Dimensions	45 x 18mm
Weight	5g

After presenting the technical specifications of the board, the main functional blocks that are important for understanding the working principle of hardware Arduino Nano are explained:

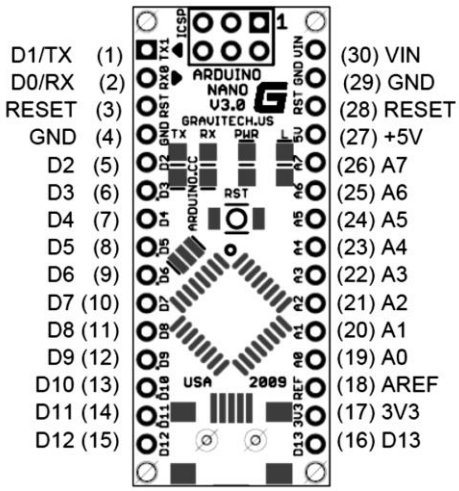

Figure 39. *Arduino Nano pinout* [53].

- **Power**: Arduino Nano can be powered via the Mini-B USB connection, 6 – 20V unregulated external power supply (pin 30) or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.
- **Memory**: the ATmega328 has 32 kB of flash memory for storing code of which 2 kB is used for the bootloader. It also has 2 kB of SRAM and 1 kB bytes of EEPROM which can be read and written with the EEPROM library.
- **Digital input / output**: each of the 14 digital pins on the Arduino Nano can be used as an input or output. They operate at 5V and can provide or receive a maximum of 40 mA and they have an internal pull-up resistor (disconnected by default) of 20 - 50 kΩ.

  In addition, some pins have specialized functions:
  - **Serial**: RX0 (pin 2) and TX1 (pin 1). Used to receive (Rx) and transmit (Tx) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL serial chip.
  - **External Interrupts**: D2 (pin 5) and D3 (pin 6). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
  - **PWM**: D3 (pin 6), D5 (pin 8), D6 (pin 9), D9 (pin 12), D10 (pin 13), and D11 (pin 14). These pins provide 8-bit PWM output.
  - **SPI**: D10 (pin 13, SS), D11 (pin 14, MOSI), D12 (pin 15, MISO), D13 (pin 16, SCK). These pins support SPI communication, which, although provided by the underlying hardware, it is not currently included in the Arduino language.
  - **LED**: there is a built-in LED connected to digital pin 13 (pin 16). When the pin is HIGH value, the LED is on, when the pin is LOW, it is off.
- **Analog inputs**: Arduino Nano has 8 analog inputs, each of which provides 10 bits of resolution (1024 different values). By default they measure from ground to 5V, though it is possible to change the upper end of their range. Analog pins A6 (pin 25) and A7 (pin 26) cannot be used as digital pins.

  Additionally, some pins have specialized functionality:
  - **I^2C**: A4 (pin 23, SDA) and A5 (pin 24, SCL). Support I^2C (Inter-Integrated Circuit) communication using the Wire library.
  - **AREF**: pin 18, it is the reference voltage for the analog inputs.
- **Communication**: Arduino Nano has facilities for communicating with a computer, another Arduino, or other microcontrollers. The microcontroller provide UART TTL (5V) serial communication, which is available on pins RX0 (pin 2) and TX1 (pin 1). An integrated circuit FTDI FT232RL on the board allows serial communication over USB and the FTDI drivers provides a virtual communication port (COM) to software on the computer. The software includes a serial monitor which allows simple textual data to be sent to and from the boards. The microcontroller ATmega328 also supports I^2C and SPI communication.

5.3 Digital-to-analog converter

To implement the slot controller, it is necessary a device that can convert the digital signal provided by Arduino into an analog signal that will supply energy to the car motor. Therefore a converter that can convert the digital signal into analog is required. The aspects to consider for proper design are explained in this section.

5.3.1 Practical operation

A digital-to-analog converter (also called DAC, D/A, D2A or D-to-A) converts an abstract finite-precision number into a physical quantity. In particular, DACs are often used to convert finite-precision time series data to a constantly varying physical signal.

A typical DAC converts the abstract numbers into a concrete sequence of impulses that are then processed by a reconstruction filter using some form of interpolation to fill in data between the impulses.

According to the Nyquist–Shannon sampling theorem [45], a digital-to-analog converter can reconstruct the original signal from the sampled data provided that its bandwidth meets certain requirements. The value of the analog output depends on a reference voltage (V_{REF}) supplied to the converter.

If the digital-to-analog converter input is a digital signal of N bits like this:

$$A_{in} = a_1 2^{N-1} + a_2 2^{N-2} + \cdots + a_{N-1} 2^1 + a_N 2^0 \quad (1)$$

where a_i denotes the bits, with 0 or 1 value, a_N denotes the leass significant bit and a_1 denotes the most significant bit.

The converter output signal is:

$$V_{out} = A_{in} \frac{V_{REF}}{2^N} \quad (2)$$

where V_{out} means the output voltage and V_{REF} means the converter reference voltage.

It has been assumed that are voltage signals, but in practice may be any type of signal. It has also been assumed that encodes a positive value. The discussion can be generalized to negative signals knowing how they are encoded.

The magnitude is defined as the output voltage variation by the least significant bit:

$$V_{LSB} = \frac{V_{REF}}{2^N} \quad (3)$$

where V_{REF} means reference voltage supplied to the converter.

It is useful, especially for the study of errors, define the dimensionless unit:

$$1\ LSB = \frac{V_{LSB}}{V_{REF}} = \frac{1}{2^N} \tag{4}$$

The following figure shows the transfer characteristic of an ideal 2 bits digital-to-analog converter. Note that the number of possible values of the converter output is 2^N, N denotes the bits of the converter. The maximum value is V_{REF} - V_{LSB} and the difference between a possible value and immediate value is V_{LSB}.

In LSB units, the maximum output value is 2^N - 1 LSB and spacing values 1 LSB.

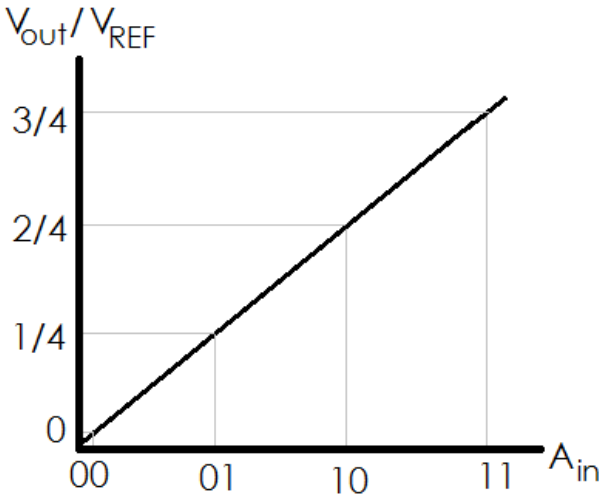

Figure 40. *2 bits digital-to-analog converter*[2].

5.3.2 Resistor ladder

There are lots of different types of digital-to-analog converter, but for this project the resistive ladder was used.

The resistor ladder or R-2R ladder is a digital-to-analog converter that uses a repeating cascaded structure composed of a precision resistor networks in a ladder-like configuration. This configuration improves the precision due to the relative ease of producing equal valued-matched resistors.

As disadvantage, for wide digital-to-analog converters is that they perform slowly due to increase large RC-constants for each added resistor link.

This digital-to-analog converter is inexpensive and relatively easy to manufacture since only two resistor values are required or only one, if R is made by placing a pair of 2R in parallel, or if 2R is made by placing a pair of R in series. It is fast and has fixed output impedance R.

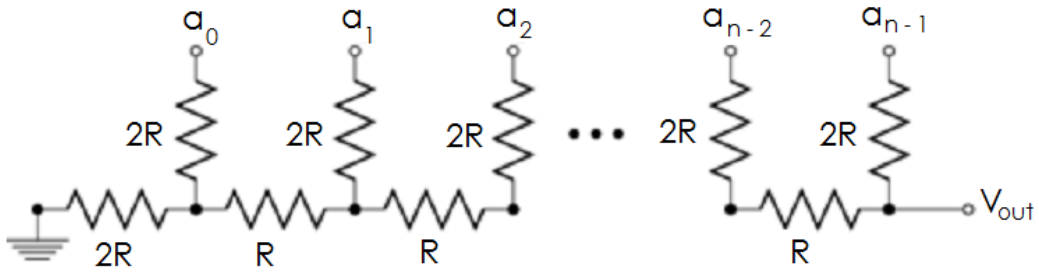

Figure 41. *Basic n-bit R-2R resistor ladder*[2].

In the example of circuit of the Figure 41, 5 bits are shown (bits a_0, a_1, a_2, a_{n-2}, a_{n-1}), giving 2^5 possible analog voltage levels at the output. The bit a_{n-1} is the most significant bit (MSB) and the bit a_0 is the least significant bit (LSB), both are driven from digital logic gates. The bits are switched between 0V (logical low 0) and V_{REF} (logical high 1). The R-2R resistor ladder causes the digital bits to be weighted in their contribution to the output voltage V_{out}.

Depending on which bits are set to logical low level and which to logical high level, the output voltage will be a corresponding stepped value between 0V and V_{REF} minus the value of the minimum step (bit a_0). The actual value of V_{REF} (and 0 volts) will depend on the type of technology used to generate the digital signals.

As explained in detail in the previous section, for a digital value A_{in}, of a resistor ladder of N bits with 0 V/V_{REF}, the output voltage Vout is:

$$V_{out} = A_{in} \frac{V_{REF}}{2^N}$$

In the previous figure shown 5 bits, N = 5 and hence $2^N = 32$. Using as an example the typical CMOS logic 1 voltage V_{REF} = 3.3 V. V_{out} will vary between 00000 with A_{in} = 0 and 11111 with A_{in} = 31.

The minimum (single step) A_{in} = 1 is:

$$V_{out} = 3{,}3 * \frac{1}{32} = 0{,}1V$$

The maximum output (11111) A_{in} = 31 is:

$$V_{out} = 3{,}3 * \frac{31}{32} = 3{,}2V$$

5.3.3 Accuracy of resistor ladder

The R-2R ladder operates as a string of current dividers whose output accuracy is dependent on how well each resistor is matched to the others. Small inaccuracies in the most significant bit resistors can entirely overwhelm the contribution of the less significant bits. Resistors used with the more significant bits must be proportionally more precise than those used with the lower significant bits.

For example, in the R-2R network shown above (Figure 41), inaccuracies in the a_{n-2} and a_{n-1} resistors must be insignificant compared to R/32 (3%). Further, to avoid problems at the 10000 to 01111 transition, the sum of the inaccuracies in the lower bits must be significantly less than R/32. The required accuracy doubles with each additional bit, for 8 bits the accuracy required will be better than 1/256 (0.4%).

On a printed circuit board, using discrete components, resistors of 1% accuracy would suffice for a 8 bit resistor ladder circuit, however with bit counts beyond this the cost of ever increasing precision resistors becomes prohibitive.

5.3.4 Buffer amplifier

For a more accurate analog output signal, a voltage buffer operational amplifier placed at the output of the resistive ladder circuit is also used.

The voltage buffer amplifier is used to transfer a voltage from the R-2R resistive ladder, having a high output impedance level, to the multifunction digital control unit with a low input impedance level. The interposed buffer amplifier prevents the multifunction digital control unit from loading the R-2R resistive ladder circuit unacceptably and interfering with its desired operation.

In the ideal voltage buffer (Figure 42), the input resistance is infinite ($Z_{in} = \infty$), the output resistance zero ($Z_{out} = 0$) because impedance of an ideal voltage source is zero. Other properties of the ideal voltage buffer are: perfect linearity, regardless of signal amplitudes and instant output response, regardless of the speed of the input signal.

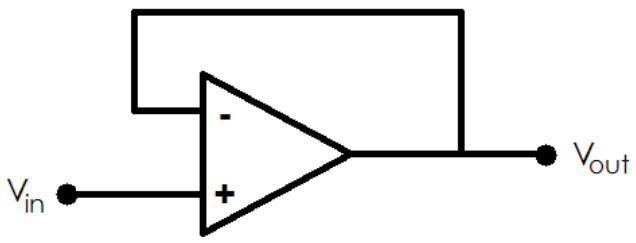

Figure 42. *Ideal voltage buffer amplifier*[2].

The voltage buffer used placed at the output of the resistive ladder circuit tracks the input voltage, so the voltage gain A_v is 1, this is a unity gain buffer; also known as a voltage follower because the voltage is transferred unchanged.

$$A_v = \frac{V_{out}}{V_{in}} = 1$$

As an example, consider a Thévenin source driving a resistor load R_L. Because of voltage division the voltage across the load is only:

$$V_R = V_A \frac{R_L}{R_L + R_A}$$

If the Thévenin source drives a unity gain voltage buffer such as that in Figure 43, the voltage input to the amplifier is V_A, and voltage division does not exist because the amplifier input resistance is infinite.

At the output, the dependent voltage source delivers voltage $V_A = A_v \cdot V_A$ to the load, again without voltage division because the output resistance of the buffer is zero. A Thévenin equivalent circuit of the combined original Thévenin source and the buffer is an ideal voltage source V_A with zero Thévenin resistance.

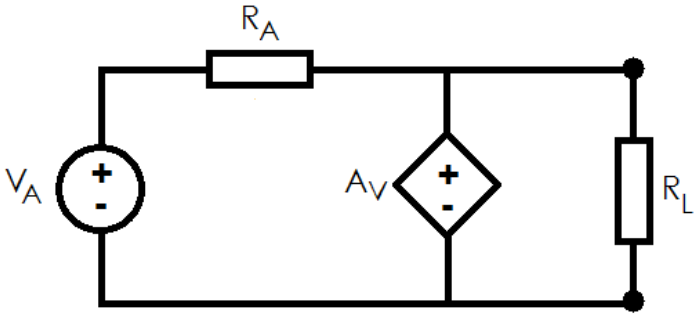

Figure 43. *Thévenin source drives a unity gain voltage buffer*[2].

5.4 Front panel

The front panel of the slot controller is formed by different hardware components that allow its proper operation. In the previous chapter the main features and basic requirements of the slot controller were listed. This chapter discusses in more detail the hardware architecture of the front panel, which is divided into multiple functional blocks that are presented below:

5.4.1 Display

The final display selected for the slot controller is a liquid-crystal display (LCD) based on thin-film (TFT) transistor technology to improve the image qualities. This display is an active-matrix LCD in where each pixel is attached to a transistor and capacitor which actively maintain the pixel state while other pixels are being addressed. Active-matrix LCD provides better contrast and addressability without being driven by circuitry, as opposed to passive-matrix LCD in which each pixel must keep its state passively.

The liquid crystal displays used in devices with passive-matrix displays have direct-driven image elements, and therefore a voltage can be easily applied across just one segment of these types of displays without interfering with the other segments. This would be impractical for a large display, because it would have a large number of colored picture elements or pixels, and thus it would require millions of connections, both top and bottom for each one of the three RGB colors of every pixel.

To avoid this issue, the pixels are addressed in rows and columns, reducing the connection count from millions down to thousands. The column and row wires attach to transistor switches, one by each pixel. The one-way current passing characteristic of the transistor prevents the charge that is being applied to each pixel from being drained between refreshes to a display's image. Each pixel is a small capacitor with a layer of insulating liquid crystal sandwiched between transparent conductive indium tin oxide (ITO) layers.

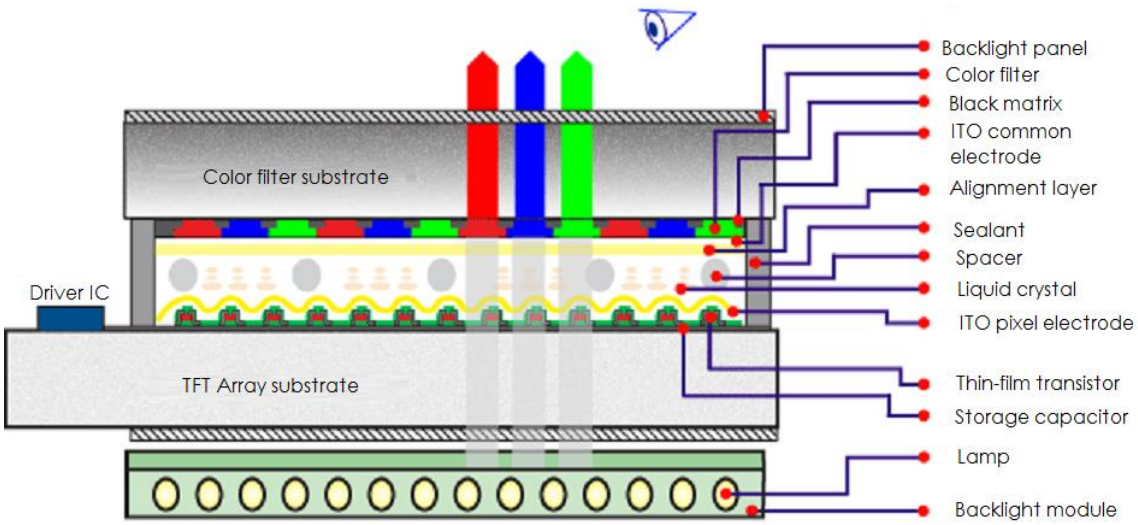

Figure 44. *Structure of thin-film-transistor liquid-crystal display* [46].

The circuit layout process of a TFT LCD is very similar to the one of semiconductor products. However, rather than fabricating the transistors from silicon, that is formed into a crystalline silicon wafer, they are made from a thin film of amorphous silicon that is deposited on a glass panel. The silicon layer for TFT LCDs is typically deposited using the plasma-enhanced chemical vapor deposition (PECVD) process. Transistors take up only a small fraction of the area of each pixel and the rest of the silicon film is etched away to allow light to easily pass through it.

Polycrystalline silicon is sometimes used in displays requiring higher TFT performance. Examples include small high-resolution displays such as those found in projectors or viewfinders. Amorphous silicon-based TFTs are by far the most common, due to their lower production cost, whereas the polycrystalline silicon TFTs are more costly and much more difficult to produce.

Arduino TFT LCD

After making a selection of different TFT LCD screens on the market and because of the features it offers and its low cost, finally the Arduino TFT LCD was chosen to implement in the slot controller.

This screen is 1.77" diagonal, with 160 x 128 pixel resolution and includes a driver that can display full 16-bit color. The red and blue have 5-bits of resolution each (32 levels of red and blue), the green has 6-bits of resolution (64 different levels). For consistency with other applications, the library deals with color in 8-bit values for the red, green, and blue channels (0-255), and scales the colors appropriately. The ST7735R is a single-chip driver for 262K-color and it consists of 396 source line and 162 gate line driving circuits. This chip is capable of connecting directly to an external microprocessor, and accepts Serial Peripheral Interface (SPI) to receive image data.

Arduino TFT LCD is a transmissive display that has a clear polarizer on the front and the back. The display therefore depends on light coming through from the back of the display toward the observer in order to be seen. There is also an onboard micro-SD card slot on the back of the screen. The display can be configured to use in two ways. One is to use an Arduino's hardware SPI interface. The other is to declare all the pins manually. There is no difference in the functionality of the screen between the two methods, but using hardware SPI is significantly faster when drawing, so in this project Serial Peripheral Interface bus is used.

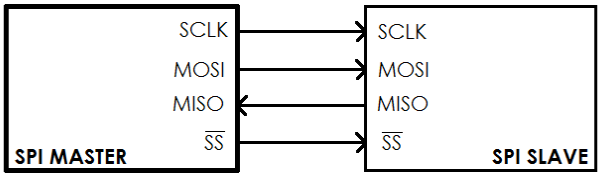

Figure 45. *Typical single SPI bus master and slave*[2].

The Serial Peripheral Interface bus is a synchronous serial communication interface specification used for short distance communication, primarily in embedded systems. SPI devices communicate in full duplex mode using a master-slave architecture with a single master. The master device originates the frame for reading and writing.

As shown in the previous Figure 45, SPI bus specifies four logic signals:

- SCLK: Serial Clock (output from master).
- MOSI: Master Output, Slave Input (output from master).
- MISO: Master Input, Slave Output (output from slave).
- SS: Slave Select (active low, output from master).

The SPI bus can operate with a single master device and with one or more slave devices. If a single slave device is used, the SS pin may be fixed to logic low.

Connections

To connect properly the Arduino TFT LCD screen (SPI slave) with the Arduino Nano platform (SPI master), the following connections were made:

Figure 46. *Arduino TFT LCD pinout* [47].

- **+5V**: this is the power pin, connect to 3.3 – 5V. It has reverse polarity protection.
- **SPI pins**: they are the pins responsible for serial communication between the display and the microcontroller.
    - **MISO**: it is the SPI Master In Slave Out pin, it is used for the SD card. It is not used for the TFT display which is write-only.
    - **SCLK**: it is the SPI clock input pin.
    - **MOSI**: it is the SPI Master Out Slave In pin, it is used to send data from the Arduino Nano microcontroller to the SD card and/or Arduino TFT LCD screen.
    - **SS1**: it is the Arduino TFT LCD SPI chip select pin.

    - **SS2**: it is the SD card chip select, used if you want to read from the SD card.
    - **D/C**: it is the SPI data or command selector pin.
- **RST**: it is the reset pin. Connected to ground to reset the TFT because it is better to have this pin controlled by the library so the display is reset cleanly.
- **BL**: it is the PWM input for the backlight control.
- **GND**: it is the power and signal ground pin

5.4.2 Push buttons

To operate the slot controller, push buttons were selected to perform the main actions that implements "Slotpiral R-evolution" because of its ease of use. Its working principle is inspired by the controller for video consoles, which allows easy handling and it is suitable for all ages.

This buttons are simple switches or electrical components that can break an electrical circuit, interrupting the current or diverting it from one conductor to another. The moving part that applies the operating force to the contacts is called the actuator.

The push buttons used in the slot controller are type of biased switch “push-to-make” also called “normally-open” or NO switch, which makes contact when the button is pressed and breaks when the button is released.

The surface is shaped to accommodate the human finger, so as to be easily depressed or pushed and the key top or head is square with different colors to differentiate each controller action.

Figure 47. *Omron push buttons with square heads*[1].

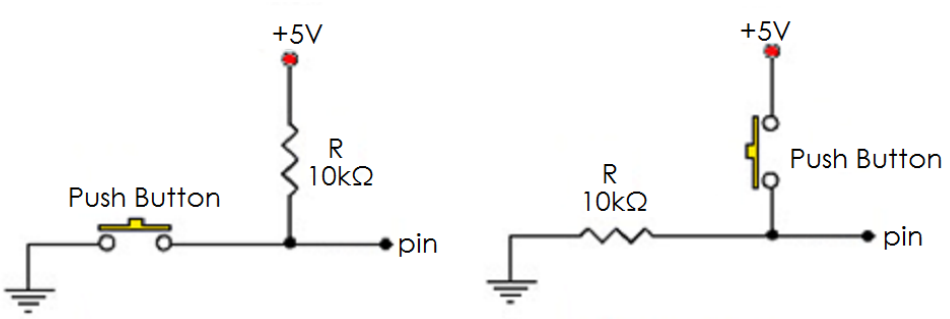

Figure 48. *Active-low (left) and active-high push button connections*[2].

Connections

There are two methods to connect the push buttons depending on the type of output:

- **Active-low**: this connection has +5V when the push button is not pressed. The pin is connected to +5V through a pull-up resistor, causing the pin to go high. But when the push button is pressed, the current from +5V is shorted to ground, causing the voltage at the pin to drop to zero. Thus, the pin is high when the push button is not pressed and low when the button is pressed.
- **Active-high**: this is the type of connection used in the project and it places +5V on the pin when the push button is pressed. The pin is connected to ground through a pull-down resistor when the push button is not pressed. Thus, the voltage at the pin is 0V. As a result, the pin is low when the push button is not pressed and high when the push button is pressed.

5.4.3 Power source

The "Slotpiral R-evolution" slot controller is divided into a transmitter module and a receiver module, so that the analysis of the power supplies is also performed separately in each module, due to each module has different requirements.

Transmitter module

The transmitter module is wireless, so you must have a power source that allows to enjoy autonomy freely.

The main energy requirements of the transmitter module are:

- **Arduino Nano**: the hardware and software platform requires an input voltage of 6 – 20V.
- **Arduino TFT LCD**: the display of the slot controller requires an input voltage of 3.3V or 5V.
- **Antenna Tx**: the antenna transmitter requires an input voltage of 3.3V or 5V.

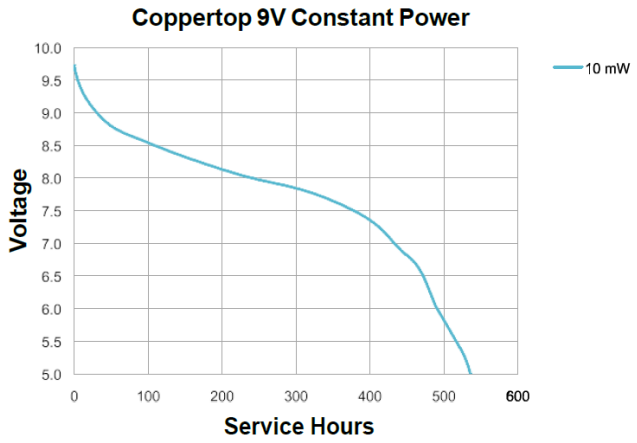

Figure 49. *Service hours of 9V battery with constant power* [48].

Given these voltage ranges required by different components of the slot controller, battery models available on the market were studied and were chosen the ones that best fit with the energy needs. Finally, the battery selected as the power source of the transmitter module was 9V battery. The slot controller is designed to work properly over the operating voltage range of a 9V battery, from fully charged (typically up to 9.6 V) to nearly dead (typically 5.0V).

These batteries are also colloquially named square battery and the dimensions are height 48.5 mm, length 26.5 mm, width 17.5 mm. Both terminals are at one end and their centers are 12.7 mm apart. Inside an alkaline or carbon-zinc 9V battery there are six cells, either cylindrical or flat type, connected in series. Some brands use welded tabs internally to attach to the cells, others press foil strips against the ends of the cells.

Rechargeable nickel–cadmium (NiCd) and nickel-metal hydride (NiMH) batteries have between six and eight 1.2 volt cells. Lithium ion versions typically use two cells (3.7V nominal each). Lithium polymer and low self-discharge NiMH versions can also be found.

Table 3. *Types of 9V batteries.*

Type	IEC name	ANSI name	Capacity (mAh)	Nominal voltage (V)	Rechargeable
Alkaline	6LR61	1604A	565	9	No
Zinc-carbon	6F22	1604D	400	9	No
Lithium	-	1604LC	1200	9.6	No
NiCd	6KR61	11604	11604	7.2	Yes
NiMH	6HR61	7.2H5	175-300	7.2	Yes
Lithium Polymer	-	-	520	7.4	Yes
Lithium-ion	-	-	620	7.4	Yes

Receiver module

The receiver module is wired, so it is not subject to restrictions on batteries, but it is necessary to analyze their electrical requirements:

- **Arduino Nano**: the processing unit platform for receiver module requires an input voltage of 6 – 20V.
- **Antenna Rx**: the antenna receiver requires an input voltage of 3.3V or 5V.
- **DAC circuits**: the electronics components of the digital-to-analog converter and the adapted circuit require an input voltage of 10V to power the integrated circuits correctly.

The multifunction digital control unit of the *Ninco* slot system used in the project has a transformer power supply that produces 14V and 3A, therefore this transformer is used as the power source for the receiver module.

The maximum voltage needed to power system is 10V, so it is necessary to reduce the 14V provided by AC/DC transformer of the multifunction digital control unit. To regulate the voltage to the proper level is used an integrated circuit LM7810 linear voltage regulator.

The LM7810 is a positive voltage regulator that produces 10V and it does not require additional components to provide a constant, regulated source of power, making it easy to use, as well as economical and efficient uses of space. This voltage regulator has three terminals (input, ground and output) and it is found in the TO220 form factor.

This integrated circuit have protections against overheating and short-circuits, making it quite robust. The input voltage must always be higher than the output voltage by minimum amount of 2.5V, so the 14V input voltage is perfect for the voltage regulator. The difference of voltage between input and output is dissipated as heat.

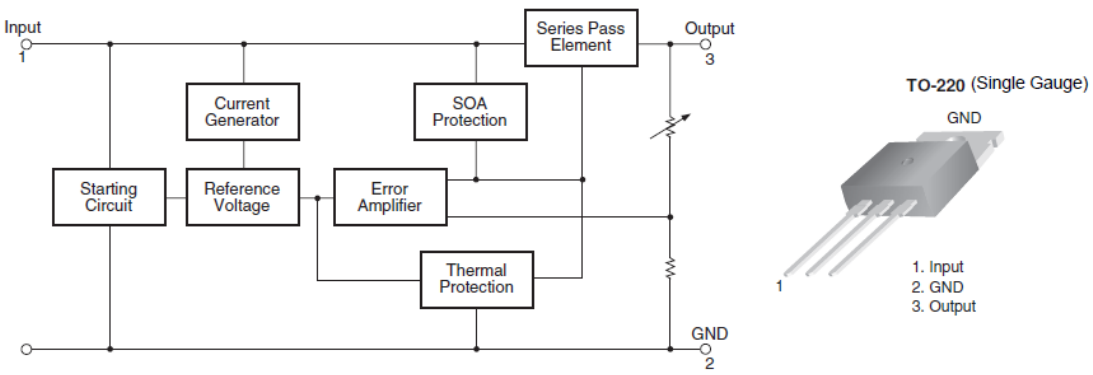

Figure 50. *Block diagram (left) and pin assignment of LM7810 [49].*

5.4.4 Communications

The communication used in the "Slotpiral R-evolution" slot controller is divided into two sections to study in detail, differentiating wired communication and wireless communication. The receiver module of the controller is based on wired communication, while wireless communication is used for the transmitter module.

Wired communication

The wired communication is implemented in the receiver module of the slot controller for connecting the multifunction digital control unit with this module. For sending and receiving information between the multifunction control unit and the controller, 4 wires are required as explained in the previous chapter, so a cable with this requirement was chosen.

Finally, the cable selected for the wired communication is a twisted pair cable. This cable is the most popular network cable and is often used in data networks for short and medium length connections due to its relatively lower costs compared to other cables.

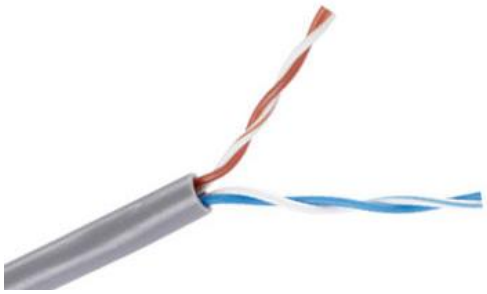

Figure 51. *Unshielded twisted pair*[1].

For wired communication of the project is not necessary to use shielded twisted pair cables, so unshielded twisted pair (UTP) cable was selected. This UTP cable consists of 4 annealed electrolytic copper wires of 0.51mm diameter. It is isolated by solid polyethylene and it has a polyvinyl chloride sheath.

Table 4. *Unshielded twisted pair technical specifications.*

Unshielded twisted pair technical specifications	
Number of pairs	2
External diameter	4.4mm
Weight	24kg/km
Resistance	93MΩ/km
Insulation resistance	8000MΩ/km

Wireless communication

The wireless communication is implemented in both modules of the slot controller for connecting the transmitter module with the receiver module. To select the type of wireless technology used for the controller the most important technologies at present were analyzed and finally a ZigBee device was chosen.

ZigBee is a specification for a suite of high-level communication protocols used to create personal area networks (PAN) built from small, low-power digital radios. ZigBee is based on an IEEE 802.15.4 standard [50] which specifies the physical layer and media access control for low-rate wireless personal area networks (LR-WPANs).

ZigBee is typically used in low data rate applications that require long battery life and secure networking. This makes it an ideal technology for this project because no big data volumes of information are transmitted, although as quickly as possible, and it is important that the security of the system is not affected by several factors.

The main features of this technology are summarized in the following points:

- **Low power consumption**: it consumes very low power and it is characterized by long battery life.
- **Long distance transmission**: limits transmission distances to 10 – 100 meters line-of-sight and ZigBee devices can transmit data over long distances by passing data through a mesh network of intermediate devices to reach more distant ones.
- **Security communication**: ZigBee networks are secured by 128 bit symmetric encryption keys.
- **Low cost technology**: it is simpler and less expensive than other wireless personal area networks (WPANs), such as Bluetooth or Wi-Fi.
- **International standard**: ZigBee operates in the industrial, scientific and medical (ISM) radio bands: 2.4GHz in most jurisdictions worldwide, 868MHz in Europe, 784MHz in China and 915MHz in the USA and Australia.
- **Support different networks**: the network layer natively supports both star and tree type networks, and generic mesh networking.

As shown in the following figure, the definition of the network layers is based on the Open Systems Interconnection model (OSI model). Only the lower layers are defined in this standard. The interaction with upper layers is intended using an IEEE 802.2 logical link control sublayer and accessing the media access control (MAC) through a convergence sublayer.

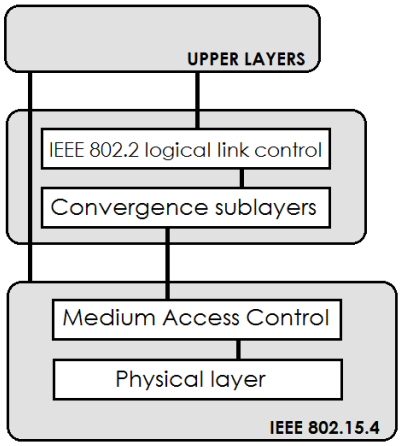

Figure 52. *IEEE 802.15.4 protocol stack*[2].

There are 3 types of ZigBee devices:

- **ZigBee Router (ZR)**: it can act as an intermediate router, passing on data from other devices.
- **ZigBee End Device (ZED)**: it cannot relay data from other devices. It only contains enough functionality to talk to the parent node.
- **ZigBee Coordinator (ZC)**: it forms the root of the network tree and might bridge to other networks. There is one ZC in each network since it is the device that started the network originally.

The radiofrequency module used tom implement wireless communication in the project is the XBee Series 1, manufactured by *Digi International* [51]. This module has many features that make it ideal for the wireless communication of the slot controller. Then a tab with its main features is presented below:

Table 5. *Specifications of the XBee Series 1.*

Specifications of the XBee Series 1	
Indoor/outdoor range	30m / 100m
Transmit power output	1mW (0 dBm)
Receiver sensitivity	-92 dBm (1% packet error rate)
Supply Voltage	2.8 - 3.4V
Transmit current	45mA (@ 3.3V)
Power-down current	< 10 µA
Operating frequency	ISM 2.4 GHz
Dimensions	2.438cm x 2.761cm
Number of channels	16 Direct Sequence Channels
Supported network topologies	Point-to-point, Point-to-multipoint & Peer-to-peer

The XBee module is designed to mount into a receptacle and therefore does not require any soldering when mounting it to a board. As shown in Figure 53, the XBee module has 20 pins but it is not necessary to connect them all to work properly. Minimum connections for the radio frequency module to transmit and receive information are:

- **VCC**: pin 1. It is connected to the power supply.
- **GND**: pin 10. It is the ground pin.
- **DOUT**: pin 2. This is an output pin responsible for UART Data Out.
- **DIN/CONFIG**: pin 3. This is an input pin responsible for UART Data In.

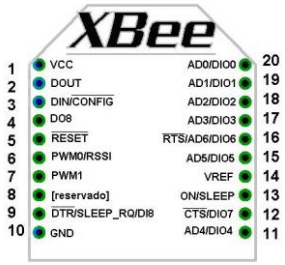

Figure 53. *XBee Series 1 pinout* [51].

The XBee modules interface to a host device through a logic-level asynchronous serial port. Through its serial port, the module can communicate with any logic and voltage compatible UART or through a level translator to any serial device, as Arduino Nano platform. Data enters the module UART through the DIN/CONFIG pin (pin 3) as an asynchronous serial signal. The signal should idle high when no data is being transmitted.

Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). The following figure illustrates the serial bit pattern of data passing through the module. Serial communications depend on the two UARTs (the Arduino Nano and the XBee module) to be configured with compatible settings like baud rate, parity, start bits, stop bits and data bits.

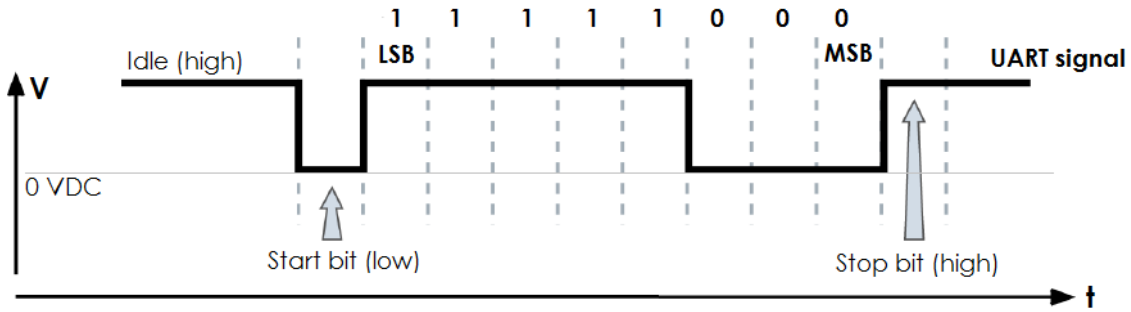

Figure 54. *UART data packet 0x1F as transmitted through the XBee*[2].

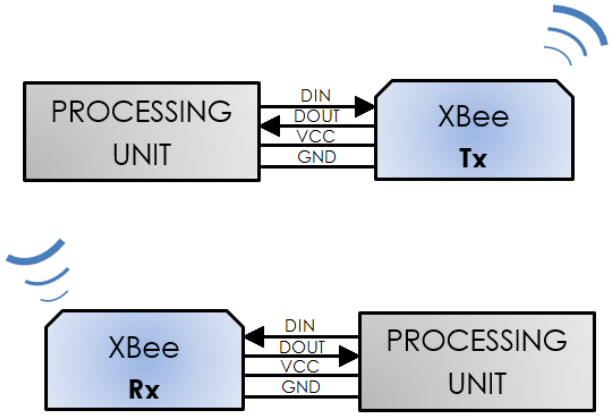

Figure 55. *System Data Flow Diagram in a UART*[2].

The radiofrequency module may operate in different modes. For its use in the transmission and reception of the slot controller is operated in transparent mode. When operating in this mode, the XBee modules act as a serial line replacement, so all UART data received through the DIN/CONFIG pin is queued up for radiofrequency transmission. When radiofrequency data is received, the data is sent out the DOUT pin.

6. SOFTWARE ARCHITECTURE

In this chapter, an analysis of the control software programmed into the microcontroller of the Arduino platform is performed.

The control algorithms used for each operating mode are explained using flowcharts. The characteristics of the different driving modes are also specified by comparative tables representing the motor power band of slot vehicles.

6.1 Introduction

This chapter discusses the software architecture of the project and it is divided into several parts for detailed study.

First, the general block of the program code and local variables common to all driving modes are explained.The algorithms used for programming the graphics controller interface and the code used to run the welcome message are also details.

From the overall structure of the program, the source code of each of the driving modes and algorithms are detailed. The differences between each mode are shown and flowcharts are used to better analyze the software operation.

6.2 Pin assignment

Before analyzing the development of the slot controller software, the pin assignment used in the Arduino Nano platform is indicated. Each pin of the microcontroller is associated with a push button that performs a specific function.

This assignment of inputs and outputs is done by the `setup()` function. This function is called when a sketch or program starts and it is used to initialize variables, pin modes, libraries, etc. The setup function will only run once, after each power up or reset of the Arduino Nano board.

The pin assignment of the transmitter module and receiver module is given below by the function `pinMode(pin,INPUT/OUTPUT)` that configures the specified pin to behave either as an input or an output.

6.2.1 Transmitter module

In the transmitter module there are five push buttons for activating the different actions implemented on the front panel.

```
void setup() {
      Serial.begin(9600); // Initialize serial communication
      pinMode(A0,INPUT); // Throttle push button (Th)
      pinMode(A1,INPUT); // Brake push button (B)
      pinMode(A2,INPUT); // Turbo push button (T)
      pinMode(A3,INPUT); // Pass push button (P)
      pinMode(A4,INPUT); // Mode push button (M)
      screen.begin();  // Initialize the screen
```

Table 6. *Pin assignment of transmitter module.*

Pin	I / O	Function
A0	Input	Throttle push button (red)
A1	Input	Brake push button (yellow)
A2	Input	Turbo push button (blue)
A3	Input	Pass push button (green)
A4	Input	Mode push button (white)

6.2.2 Receiver module

In the receiver module there is a push button for activating the random pass algorithm when the automatic driving mode is used.

```
void setup() {
      Serial.begin(9600); // Initialize serial communication
      for(int i=2;i<=9;i++){pinMode(i,OUTPUT);} // Bits DAC D2-D9
      pinMode(12,OUTPUT); // LED pass + high pulse
      pinMode(11,OUTPUT); // LED automatic mode
      pinMode(10,OUTPUT); // Relay coil
      pinMode(A0,INPUT); // Receiver mode push button (M)
```

Table 7. *Pin assignment of receiver module.*

Pin	I / O	Function
D2 - D9	Output	8 bits serial data
D12	Output	LED pass and high pulse
D11	Output	LED automatic mode
D10	Output	Relay coil
A0	Input	Mode push button (white)

6.3 Transmitter software

In this section, the transmitter module software is explained in detail, starting from the general structure of the program, local variables and libraries, also the programming code of each driving mode of the slot controller is analyzed.

6.3.1 General structure

The following figure is a flowchart of the overall structure of the transmitter module program.

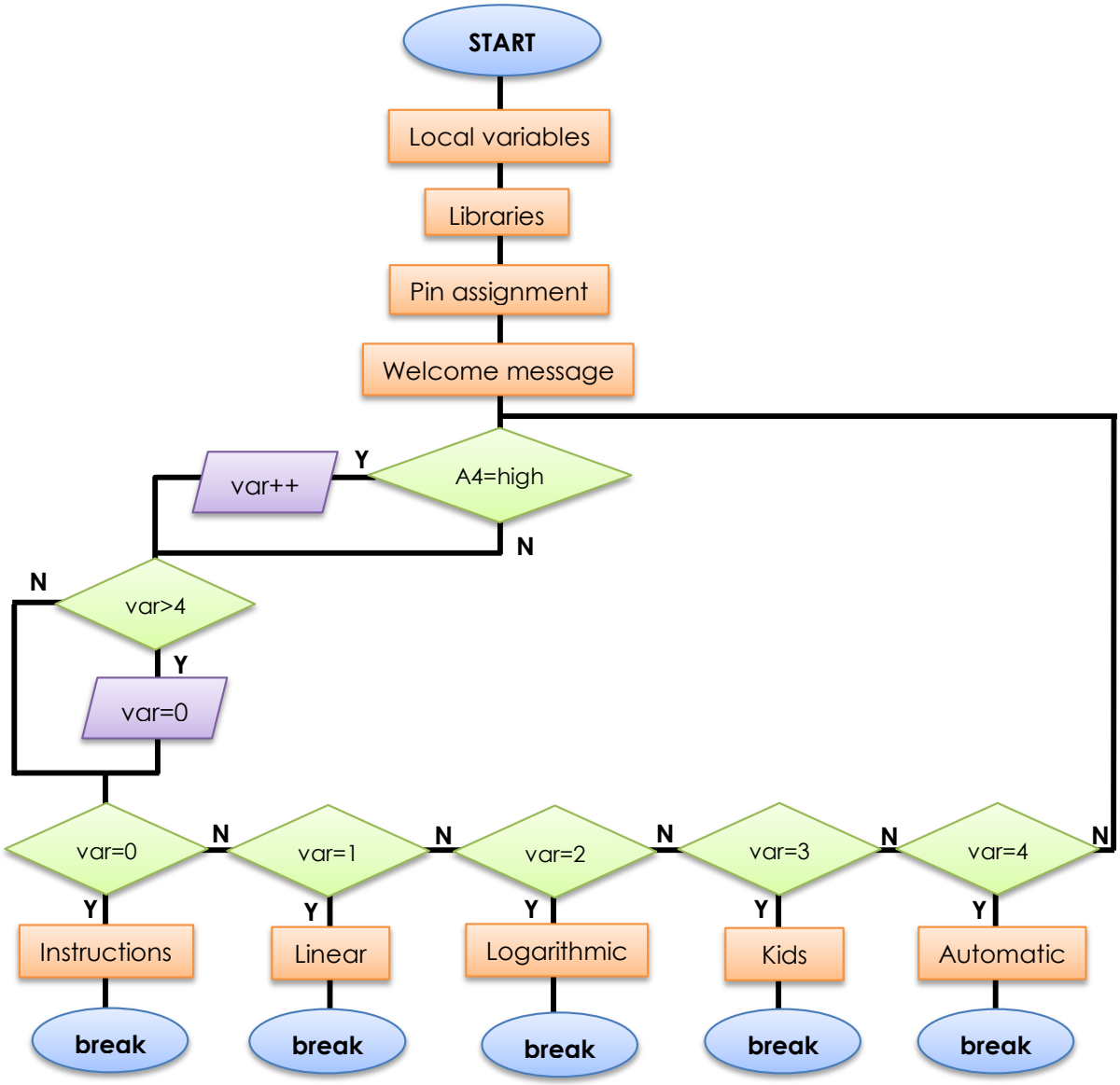

Figure 56. *General flowchart of the transmitter software.*

The basic program for the previous flowchart, without detailing the program included in each `case` statement is:

```
void loop() {
      Serial.write(numero);
      if(digitalRead(A4)==HIGH){ // Mode selection
            delay(500);
            var++;
            if(var>4){var=0;}}
      switch (var) {
      case 0: // INSTRUCTIONS
      ...
      Break;
      case 1: // MODE LINEAR
      ...
      Break;
      case 2: // MODE LOGARITHMIC
      ...
      Break;
      case 3: // MODE KIDS
      ...
      Break;
      case 4: // MODE AUTOMATIC
      ...
      Break;
      } // End switch structure
```

After creating a `setup()` function, which initializes and sets the initial values, the `loop()` function loops consecutively, allowing the program to change and respond.

The structure used for the selection of the driving modes of the slot controller is the `switch / case` statement. This statement controls the flow of the program by allowing specifying a different code that should be executed in each driving mode. In particular, the `switch` statement compares the value of the variable (`var`) to the values specified in `case` statements, corresponding to the driving modes. When a `case` statement is found whose value matches that of the variable, the code in that `case` statement is run.

The `break` statement exits the `switch` statement, and it is used at the end of each `case`, because without a `break`, the `switch` statement will continue executing the following expressions until a `break` or the end of the `switch` statement is reached.

6.3.2 Global variables

A global variable is one that can be seen by every function in the program code. When programs start to get complex, variables are a useful way to insure that only one function has access to its own variables. This prevents programming errors when one function inadvertently modifies variables used by another function.

The global variables used in the program code of “Slotpiral R-evolution” transmitter are shown on the next page.

```
int stlin=14; // Throttle steps linear mode
int sblin=38; // Brake steps linear mode
int stlog=10; // Throttle steps logarithmic mode
int sblog=38; // Brake steps logarithmic mode
float n=1; // Formula steps logarithmic mode
int stkid=10; // Throttle steps kids mode
int sbkid=30; // Brake steps kids mode
int staut=10; // Throttle steps automatic mode
int sbaut=10; // Brake steps automatic mode
int num=255; // Counter initialize min speed
int var=0; // Switch case structure variable
int ctlog=147; // Logarithmic mode constant
int cte=147; // Mode logarithmic constant
int pass=171; // Pass constant
int a=0; // Graphical control variables
int b=0;
int c=0;
```

Most global variables created are `int` type and they are refered to increases or decreases steps in the `num` variable depending on the driving mode selected. Integers (`int`) are the primary data-type for number storage. These different steps of increasing and decreasing are important because they form the engine power curve as explained in the following sections.

6.3.3 Libraries

```
#include <SPI.h> // SPI communication library
#include <TFT.h> // Arduino TFT library
#define cs   10 // Define screen CS pin
#define dc   9 // Define screen DC pin
#define rst  8 // Define screen RESET pin
TFT screen = TFT(cs, dc, rst);
```

The TFT library enables an Arduino board to communicate with the Arduino TFT LCD screen. This library extends the Adafruit GFX, and Adafruit ST7735 libraries that it is based on. The GFX library is responsible for the drawing routines, while the ST7735 library is specific to the screen on the Arduino TFT. The Arduino specific additions were designed to work as similarly to the Processing API as possible.

The TFT library relies on the SPI library for communication with the screen and allows communicating with Serial Peripheral Interface (SPI) devices, with the Arduino Nano as the master device. If using hardware SPI is necessary to declare the CS, DC, and RESET pins, as MOSI and SCLK which are already defined.

6.3.4 Welcome message

The welcome message is a graphic animation that shows the name of the project author and universities who have contributed to its development. The program code and the functions used are listed on the next page.

```
screen.background(0,0,0);  // Background color
screen.stroke(255,128,0); // Text color
screen.text("<-SLOTPIRAL R-EVOLUTION->", 5, 3);
screen.fill(255,255,255); // Fill color
for(int i=0; i<13; i++){screen.rect(0, 123-10*i, 5, 5);}
for(int i=0; i<12; i++){screen.rect(155, 118-10*i, 5, 5);}
delay(200);
screen.rect(0, 123, 5, 5);
...
screen.text("CREDITS:", 55, 20);
screen.rect(15, 118, 5, 5);
delay(200);
...
screen.text("Santiago Ros Navarro", 20, 40);
screen.rect(30, 123, 5, 5);
delay(200);
...
screen.text("Technical University of", 10, 60);
screen.text("Cartagena (UPCT)", 35, 70);
screen.rect(45, 118, 5, 5);
delay(200);
...
screen.text("Tampere University of", 15, 90);
screen.text("Technology (TUT)", 35, 100);
screen.rect(60, 123, 5, 5);
delay(200);
...
delay(1000);
screen.background(0,0,77);  // Refresh background
```

The function `screen.rect(xStart, yStart, width, height)` draws a rectangle to the TFT screen with 4 arguments, the first two are at the top left corner of the shape, the last two are at the width and height of the shape.

To erase everything currently on the LCD screen with a determinate color ranged between 0-255 is used the function `screen.background(R, G, B)`.

The text is programmed by the function `screen.text(text, xPos, yPos)`, that writes text to the screen at the given coordinates.

With `screen.fill(R, G, B)` the fill color of rectangles are set and the function `screen.stroke(R, G, B)` sets the color of borders around rectangles and text.

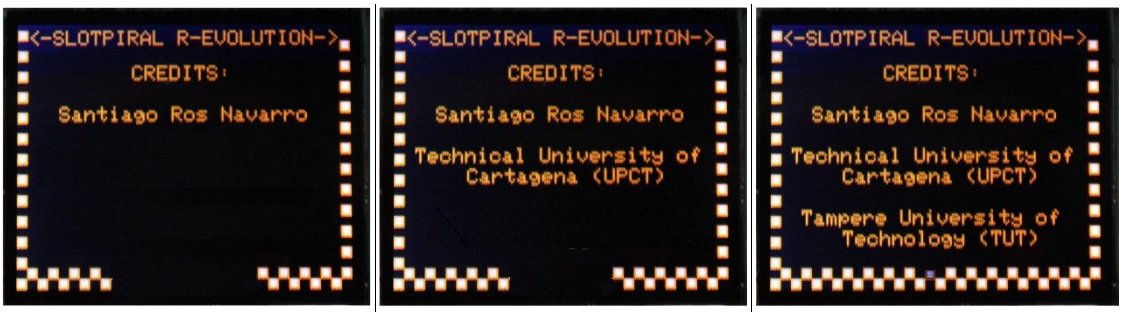

Figure 57.*Welcome message screenshot.*

6.3.5 Instructions

The instructions message is a static animation that shows the name of the different driving modes available in the slot controller and the order of these modes on the screen. The graphical representation of a vehicle traveling on a road with the race number 7 and a checkered flag also appear on the screen.

The programming code used to program the instructions section is shown below:

```
case 0: // INSTRUCTIONS_____________________________________________________
  if(c==0){
  num=255; // Refresh speed residual from automatic mode
  screen.background(0,0,77);
  screen.stroke(255,128,0);
  screen.text("<-SLOTPIRAL R-EVOLUTION->", 5, 3);
  screen.text("Press [M] to select mode:", 5, 20);
  screen.text("1. Linear ", 10, 35);
  screen.text("2. Logarithmic ", 10, 50);
  screen.text("3. Kids ", 10, 65);
  screen.text("4. Automatic ", 10, 80);
  screen.fill(255,255,255); // Flag checkered
  for(int i=0; i<8; i++){screen.rect(10*i, 123, 5, 5);}
  for(int i=0; i<8; i++){screen.rect(5+10*i, 118, 5, 5);}
  for(int i=0; i<9; i++){screen.rect(10*i, 113, 5, 5);}
  for(int i=0; i<8; i++){screen.rect(5+10*i, 108, 5, 5);}
  for(int i=0; i<9; i++){screen.rect(10*i, 103, 5, 5);}
  screen.fill(0,0,0); // Road
  screen.circle(160, 124, 80);
  screen.fill(0,0,77);
  screen.circle(160, 124, 27);
  screen.fill(255,255,255); // Flag checkered
  for(int i=0; i<3; i++){screen.rect(135+10*i, 123, 5, 5);}
  for(int i=0; i<2; i++){screen.rect(140+10*i, 118, 5, 5);}
  for(int i=0; i<3; i++){screen.rect(135+10*i, 113, 5, 5);}
  for(int i=0; i<2; i++){screen.rect(140+10*i, 108, 5, 5);}
  for(int i=0; i<2; i++){screen.rect(145+10*i, 103, 5, 5);}
  screen.fill(255,0,0); // Red car
  screen.noStroke();
  screen.rect(105, 88, 20, 33);
  screen.fill(150,150,150);
  screen.rect(103, 94, 3, 5);
  screen.rect(103, 111, 3, 5);
  screen.rect(124, 94, 3, 5);
  screen.rect(124, 111, 3, 5);
  screen.stroke(255,255,255);
  screen.fill(255,0,0);
  screen.circle(115, 105, 6);
  screen.text("7", 113, 102);
  screen.stroke(177,0,0);
  screen.line(108, 96, 122, 96);
  screen.line(108, 114, 122, 114);
  screen.stroke(255,128,0); }
  c=1; // Graphical control variable disable
break;
```

Figure 58. *Instructions screenshot.*

The function `screen.circle(xPos, yPos, radius)` draws a circle on the screen, which is used to draw the road. This circle is drawn relative to its center point, which means the total diameter.

After calling the function `screen.noStroke()`, any shapes drawn on screen will not have an outline stroke color.

6.3.6 Linear mode

The first driving mode available to the slot controller is the linear mode. This mode is the easiest to control and it is based on an arithmetic progression of the variable `num` for acceleration algorithm and braking. Table 8 shows the main characteristics of the control variable. The acceleration and deceleration steps refer to the value that is increased or decreased this variable.

The maximum speed value is the value of the variable `num` which provides the maximum voltage to the slot vehicle motor, which is 1.35V or 3.7V electric potential difference, as explained in chapter 4. The zero speed value is the value of the variable `num` without supplies voltage to the slot motor vehicle, which is 0V or 5.05V electric potential difference.

Table 8. *Characteristics of the control variable in the linear mode.*

Characteristics of `num` variable in linear mode	
Maximum speed / turbo value	64
Zero speed value	255
Pass speed value	171
Total speed (%)	100%
Acceleration step (`stlin`)	14
Deceleration step (`sblin`)	38

The following figure is a flowchart of the linear driving mode of the slot controller:

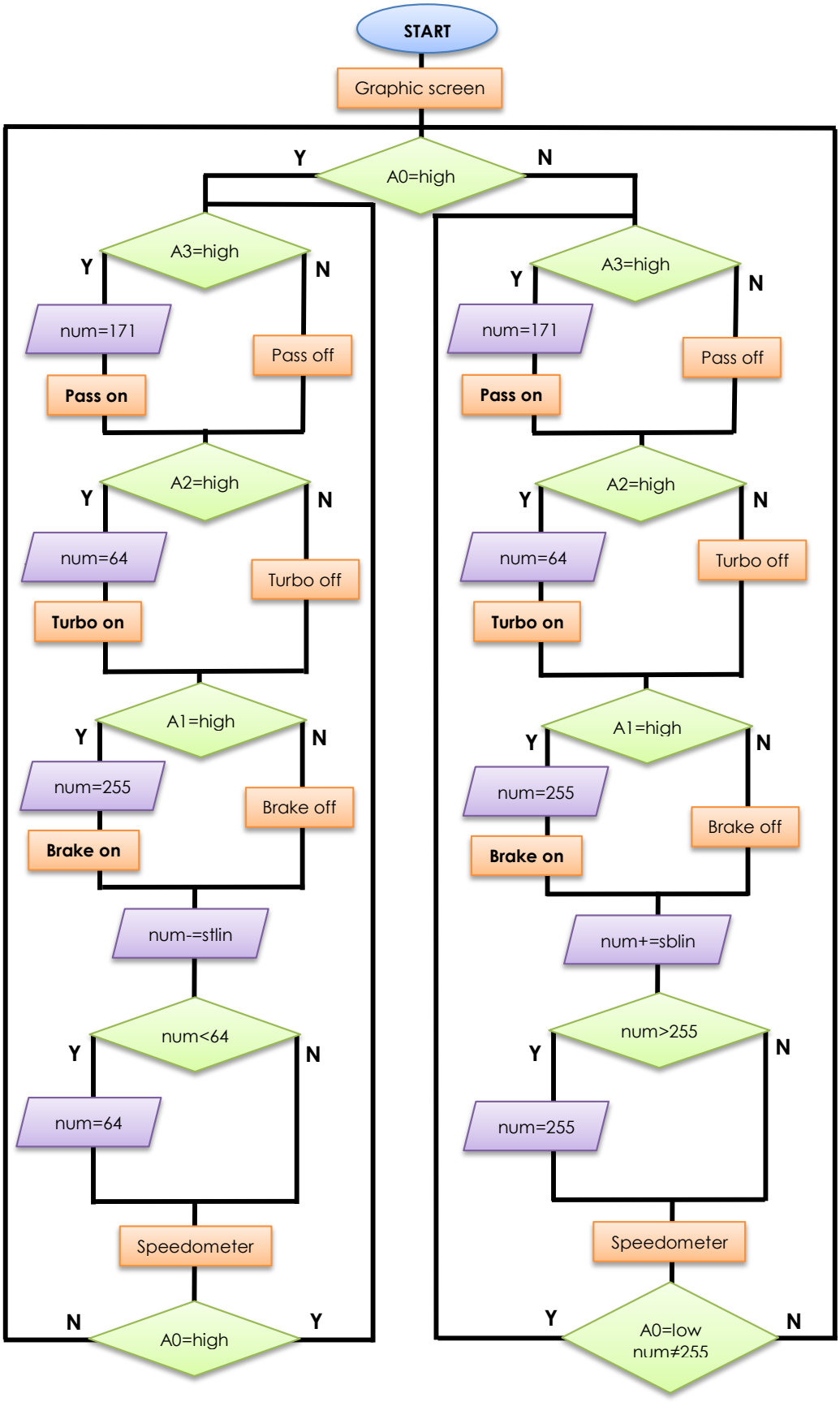

Figure 59. *Flowchart of the linear mode.*

The previous flowchart shows the existence of a `while` loop in which there is contained a `do-while` loop that runs the algorithm acceleration, reducing the variable `num` as an arithmetic progression.

For deceleration, there is another structure `do-while` loop that is responsible for reducing the speed of the slot vehicle, incrementing the variable `num` according to another arithmetic progression.

This mode is the beginning for programming the different driving modes offered by the "Slotpiral R-evolution" controller. Thereupon, the programming code of the linear mode is explained in detail and the other driving modes are presented in the following sections, discussing the main differences in respect of the linear mode.

First, the programming code responsible for drawing the information from the TFT LCD screen is run only once. This code draws the axes of the graph of speed versus time and the silhouette of the speedometer. The data is sent to the receiver module using the `Serial.write(num)` function.

```
case 1: // MODE LINEAR_____________________________________________________
  Serial.write(num);
  if(a==0){screen.background(0,0,77); // Graphic screen
    screen.fill(64,64,64);
    screen.rect(0, 0, 160, 13);
    screen.stroke(100,100,100);
    screen.circle(25, 135, 27);
    screen.circle(80, 135, 30);
    screen.circle(135, 135, 27);
    screen.stroke(255,128,0);
    screen.text("<-SLOTPIRAL R-EVOLUTION->", 5, 3);
    screen.stroke(255,255,255);
    screen.line(15, 90, 145, 90); // Axis X
    screen.line(15, 89, 145, 89);
    screen.text("t", 152, 86);
    screen.line(14, 91, 14, 18); // Axis Y
    screen.line(15, 18, 15, 90);
    screen.text("v", 4, 18);
    screen.text("Mode: linear", 40, 93);
    a=1; // Graphical control variable disable
    b=0; // Graphical control variable enable
    screen.noStroke(); // Speedometer silhouette
    screen.fill(64,64,64);
    for(int i=0; i<13; i++){screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
  }
```

Then, the program structure is divided into two main blocks. The first block is responsible for acceleration and it is composed by a `while` loop containing another `do-while` structure inside. In this `do-while` loop, the program checks if the pass, turbo o brake push buttons are pressed using `if-else` structures that allow these actions to take precedence over the acceleration when several push buttons are pressed simultaneously.

In the `if-else` structures responsible for the actions of pass and brake must be added the residual value of the global variable `stlin` for the algorithm to run properly.

The `do-while` loop increases the vehicle speed, decreasing the variable `num` through an arithmetic progression, reaching the value of maximum speed and let it fixed. The data is also sent to the receiver module using the `Serial.write(num)` function. Finally, the programming code that shows the vehicle speed on the graphic display is run and continues running the code while the throttle push button is pressed.

```
while(digitalRead(A0)==HIGH){ // Acceleration
  do{
    if(digitalRead(A3)==HIGH){ // Pass
      digitalWrite(7,HIGH);
      num=171+stlin; // Pass constant + residual throttle
      screen.stroke(255,255,0);
      screen.text("PASS", 67, 116);
      screen.noStroke();
    for(int i=0; i<7; i++){
      screen.fill(255,255-17*i,0);
      screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
    for(int i=7; i<13; i++){
      screen.fill(64,64,64);
      screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
    }else{digitalWrite(7,LOW);
      screen.stroke(102,102,0);
      screen.text("PASS", 67, 116);
      screen.noStroke();}
    if(digitalRead(A2)==HIGH){ // Turbo
      num=64;
      screen.stroke(255,0,0);
      screen.text("TURBO", 120, 116);
      screen.noStroke();
      for(int i=0; i<13; i++){
        screen.fill(255,255-17*i,0);
        screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
    }else{screen.stroke(102,0,0);
      screen.text("TURBO", 120, 116);
      screen.noStroke();}
    if(digitalRead(A1)==HIGH){ // Brake
      num=255+stlin; // Stop + residual throttle
      screen.stroke(0,255,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();
      screen.fill(64,64,64);
    for(int i=0; i<13; i++){screen.rect(16+i*9, 85-5*i, 10, 4+5*i);} //
Speedometer silhouette
    }else{screen.stroke(0,102,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();}

    num-=stlin; // Increase speed
    delay(100); // Regule speed
    if(num<64){num=64;} // Maximum speed fixed

    Serial.write(num); // Speedometer
    if(num<=255&&num>242){screen.fill(255,255,0); screen.rect(16, 85, 10, 4);}
```

```
  if(num<=242&&num>229){screen.fill(255,238,0); screen.rect(25, 80, 10, 9);}
  if(num<=229&&num>216){screen.fill(255,221,0); screen.rect(34, 75, 10, 14);}
  if(num<=216&&num>203){screen.fill(255,204,0); screen.rect(43, 70, 10, 19);}
  if(num<=203&&num>190){screen.fill(255,187,0); screen.rect(52, 65, 10, 24);}
  if(num<=190&&num>177){screen.fill(255,170,0); screen.rect(61, 60, 10, 29);}
  if(num<=177&&num>164){screen.fill(255,153,0); screen.rect(70, 55, 10, 34);}
  if(num<=164&&num>151){screen.fill(255,136,0); screen.rect(79, 50, 10, 39);}
  if(num<=151&&num>138){screen.fill(255,119,0); screen.rect(88, 45, 10, 44);}
  if(num<=138&&num>125){screen.fill(255,102,0); screen.rect(97, 40, 10, 49);}
  if(num<=125&&num>112){screen.fill(255,85,0); screen.rect(106, 35, 10, 54);}
  if(num<=112&&num>99){screen.fill(255,68,0); screen.rect(115, 30, 10, 59); }
  if(num<=99&&num>63){screen.fill(255,51,0); screen.rect(124, 25, 10, 64);}
}while(digitalRead(A0)==HIGH); // End do-while structure
} // End while structure
```

The block program corresponding to the deceleration is another `do-while` loop independent and also contains algorithms to check if the pass, turbo o brake push buttons are pressed using `if-else` structures that allow these actions to take precedence over the acceleration when several push buttons are pressed simultaneously.

In the `if-else` structures responsible for the actions of pass and turbo must be subtracted the residual value of the global variable `sblin` for the algorithm to run properly. The residual value of the variable `stlin` must also be added to the `if-else` loop responsible for braking.

The `do-while` loop decreases the vehicle speed, increasing the variable `num` through an arithmetic progression, reaching the value of zero speed and let it fixed. The data is sent to the receiver module using the `Serial.write(num)` function. The programming code that shows the vehicle speed in real time on the graphic display is also run. Finally, the word `break` is used to exit from the switch statement.

```
do{ // Deceleration
  if(digitalRead(A3)==HIGH){ // Pass
    digitalWrite(7,HIGH);
    num=171-sblin; // Pass constant - residual brake
    screen.stroke(255,255,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();
    for(int i=0; i<7; i++){
      screen.fill(255,255-17*i,0);
      screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
    for(int i=7; i<13; i++){
      screen.fill(64,64,64);
      screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
  }else{digitalWrite(7,LOW);
    screen.stroke(102,102,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();}
  if(digitalRead(A2)==HIGH){ // Turbo
    num=64-sblin; // Maximum speed - residual brake
    screen.stroke(255,0,0);
    screen.text("TURBO", 120, 116);
    screen.noStroke();
    for(int i=0; i<13; i++){
```

```
        screen.fill(255,255-17*i,0);
        screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
    }else{screen.stroke(102,0,0);
      screen.text("TURBO", 120, 116);
      screen.noStroke();}
    if(digitalRead(A1)==HIGH){ // Brake
      num=255+stlin; // Stop + residual throttle
      screen.stroke(0,255,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();
      screen.fill(64,64,64);
      for(int i=0; i<13; i++){screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
      }else{screen.stroke(0,102,0);
        screen.text("BRAKE", 10, 116);
        screen.noStroke();}

    num+=sblin; // Decrease speed
    delay(100); // Regule brake
    if(num>255){num=255;} // Stop fixed

    Serial.write(num); // Speedometer
    if(num==255){screen.fill(64,64,64); screen.rect(16, 85, 10, 4);}
    ...
    if(num<=112&&num>90){screen.fill(64,64,64); screen.rect(124, 25, 10, 64);}
  }while(digitalRead(A0)==LOW&&num!=255); // End do-while structure
break;
```

Successive driving modes are explained in less detail than this linear mode, because the operation is similar once showed their differences. The complete programming code used in the slot controller is attached in the Appendix A (p. 128) and the Appendix B (p. 142) at the end of this Master's Thesis.

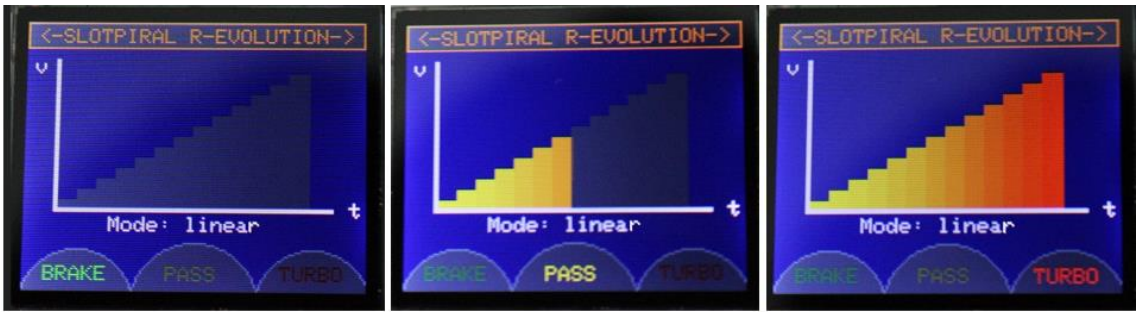

Figure 60. *Brake, pass and turbo actions screenshots in linear mode.*

6.3.7 Logarithmic mode

The second driving mode available to the slot controller is the logarithmic mode. This mode is based on a logarithmic function of the variable `num` for acceleration algorithm and an arithmetic progression of the variable `num` for brake algorithm. The figure on the next page is a flowchart of the logarithmic driving mode of the slot controller for understanding how the program works.

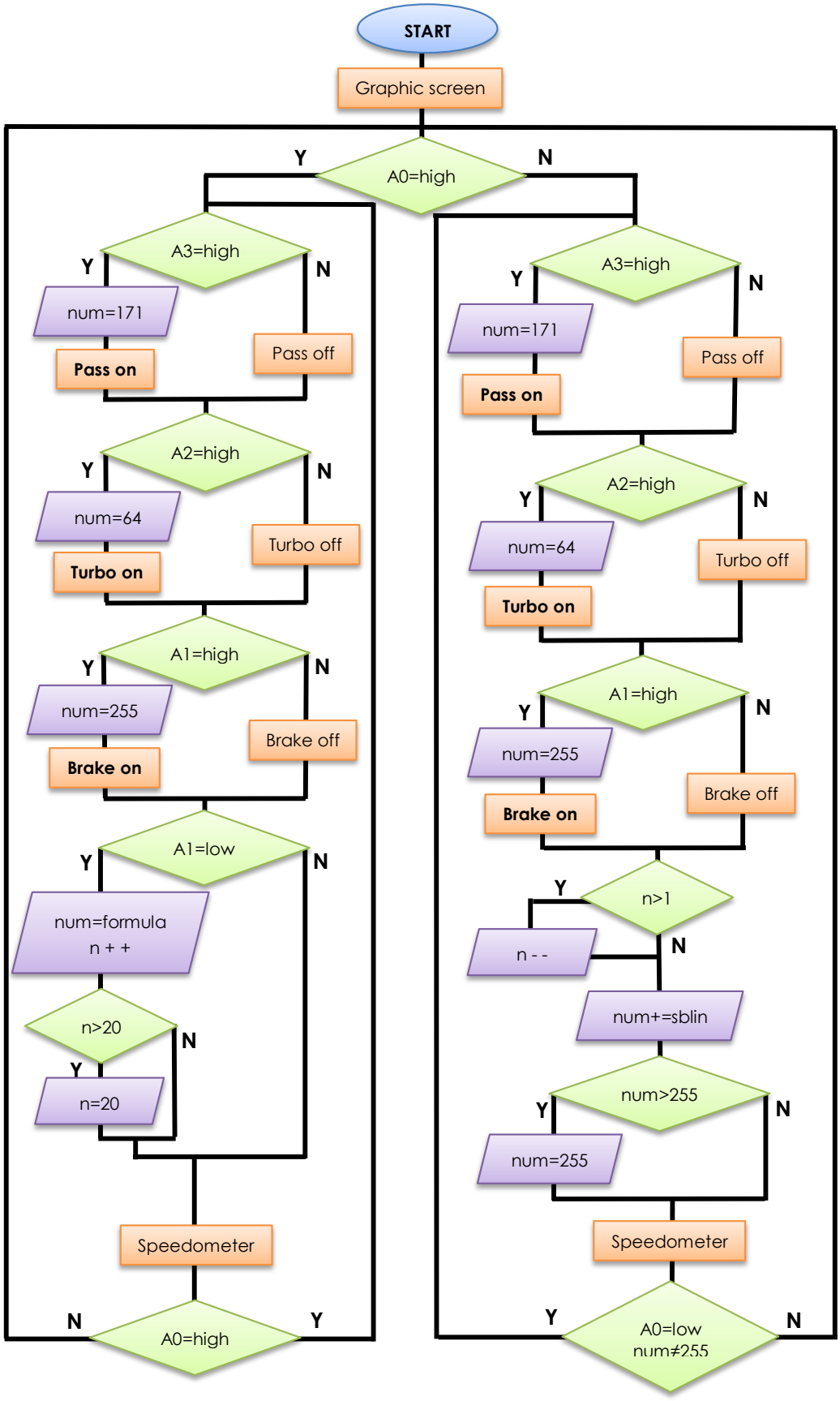

Figure 61. *Flowchart of the logarithmic mode.*

Table 9. *Characteristics of the control variable in the logarithmic mode.*

Characteristics of `num` variable in logarithmic mode	
Maximum speed / turbo value	64
Zero speed value	255
Pass speed value	171
Total speed (%)	100%
Acceleration formula	num = int {255 - [cte * $\log_{10}$ (n)] }
Constant (cte)	147
Acceleration step (`stlog`)	10
Deceleration step (`sblog`)	38

The previous flowchart is similar to the linear mode, but differs in that the increase of the variable `num` is not linear because it follows a logarithmic curve. Table 9 shows the main characteristics of the `num` control variable.

In this driving mode, the acceleration is performed by a logarithmic mathematical algorithm explained below. The rest of the software is similar to the linear mode discussed in the previous section, so this code program is not explained in detail.

The program structure is divided into two main blocks. The first block is responsible for acceleration and is composed by a `while` loop containing another `do-while` structure inside. In this `do-while` loop, the program checks if the pass, turbo o brake push buttons are pressed using `if-else` structures that allow these actions take precedence over the acceleration when several push buttons are pressed simultaneously.

In the `if-else` structures responsible for the actions of pass and brake must be added the residual value of the global variable `stlog` for the algorithm to run properly.

In the `do-while` loop there is an `if` loop that increases the slot vehicle speed, decreasing the variable `num` through a logarithmic formula divided into 20 steps with the variable `n`, reaching the value of maximum speed and let it fixed. The algorithm implemented allows to have for a range of values to fall within the permissible range (255 – 64) of the variable `num`.

$$num = int\{255 - [cte * log_{10}(n)]\} \quad (1)$$

where num denotes the variable, int converts the value to the int data type, cte denotes the constant and n means the number of logarithmic steps.

To calculate the value of the constant, with the slot vehicle at the maximum speed allowed (step 20) without exceeding the range:

$$num_{\max speed} = 64 = int\{255 - [cte * log_{10}(20)]\} \quad (2)$$

$$\frac{191}{log_{10}(20)} = int\{cte\} \quad (3)$$

$$cte = int\{146.8067\} = \mathbf{147} \quad (4)$$

To verify that it works correctly it is calculated the constant to zero speed (step 1):

$$num_{\text{zero }speed} = 255 = int\{255 - [147 * log_{10}(1)]\} = \mathbf{255} \quad (5)$$

Finally, the programming code that shows the vehicle speed on the graphic display is run and continues running the code while the throttle push button is pressed.

```
while(digitalRead(A0)==HIGH){ // Acceleration
do{
  if(digitalRead(A3)==HIGH){ // Pass
    digitalWrite(7,HIGH);
    n=3.7; // Pass logarithmic steps
    num=171+stlog; // Pass constant + residual throttle
    screen.stroke(255,255,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();
    for(int i=1; i<8; i++){
      screen.fill(255,255-17*i,0);
      screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
    for(int i=7; i<14; i++){
      screen.fill(64,64,64);
      screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
  }else{digitalWrite(7,LOW);
    screen.stroke(102,102,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();}
  if(digitalRead(A2)==HIGH){ // Turbo
    num=64;
    screen.stroke(255,0,0);
    screen.text("TURBO", 120, 116);
    screen.noStroke();
  for(int i=1; i<14; i++){
    screen.fill(255,255-17*i,0);
    screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
  }else{screen.stroke(102,0,0);
    screen.text("TURBO", 120, 116);
    screen.noStroke();}
  if(digitalRead(A1)==HIGH){ // Brake
    n=1; // Initialize logarithmic steps
    num=255+stlog; // Stop + residual throttle
    screen.stroke(0,255,0);
    screen.text("BRAKE", 10, 116);
    screen.noStroke();
```

```
      screen.fill(64,64,64);
      for(int i=1; i<14; i++){screen.rect(7+i*9, 82-int(51*log10(i)), 10,
7+int(51*log10(i)));}
    }else{screen.stroke(0,102,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();}

    if(digitalRead(A1)==LOW){
      num=int(255-(cte*log10(n))); // Increase speed
      //delay(100); // Regule brake
      n++;
      if(n>20){n=20;}} // Maximum limit speed

    Serial.write(num);
    if(num==255){screen.fill(255,255,0); screen.rect(16, 82, 10, 7);}
    ...
    if(num<=70&&num>63){screen.fill(255,51,0); screen.rect(124, 26, 10, 63);}
  } while(digitalRead(A0)==HIGH); // End do-while structure
  } // End while
```

The block program corresponding to the deceleration is another `do-while` loop independent and also contains algorithms to check if the pass, turbo o brake push buttons are pressed using `if-else` structures that allow these actions take precedence over the acceleration when are pressed several push buttons simultaneously.

In the `if-else` structures responsible for the actions of pass and turbo must be subtracted the residual value of the global variable `sblog` for the algorithm to run properly. The residual value of the variable `sblog` must also be added to the `if-else` loop responsible for braking.

The `do-while` loop decreases the vehicle speed, increasing the variable `num` through an arithmetic progression, reaching the value of zero speed and let it fixed. There is an `if` loop responsible for reducing the variable logarithm of the number of steps (`n`) to match the deceleration in real time.

The programming code that shows the vehicle speed in real time on the graphic display is also run. Finally, the word `break` is used for exit from the switch statement.

```
  do{ // Deceleration
  if(digitalRead(A3)==HIGH){ // Pass
    digitalWrite(7,HIGH);
    n=4; // Pass logarithmic steps
    num=171-sblog; // Pass constant - residual brake
    screen.stroke(255,255,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();
    for(int i=1; i<8; i++){
      screen.fill(255,255-17*i,0);
      screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
    for(int i=7; i<14; i++){
      screen.fill(64,64,64);
      screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
  }else{digitalWrite(7,LOW);
```

```
    screen.stroke(102,102,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();}
  if(digitalRead(A2)==HIGH){ // Turbo
    num=64-sblog; // Maximum speed - residual brake
    screen.stroke(255,0,0);
    screen.text("TURBO", 120, 116); screen.noStroke();
  for(int i=1; i<14; i++){
    screen.fill(255,255-17*i,0);
    screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
  }else{screen.stroke(102,0,0);
    screen.text("TURBO", 120, 116);
    screen.noStroke();}
  if(digitalRead(A1)==HIGH){ // Brake
    n=1; // Initialize logarithmic steps
    num=255+stlog; // Stop + residual throttle
    screen.stroke(0,255,0);
    screen.text("BRAKE", 10, 116);
    screen.noStroke();
    screen.fill(64,64,64);
    for(int i=1; i<14; i++){screen.rect(7+i*9, 82-int(51*log10(i)), 10,
7+int(51*log10(i)));}
  }else{screen.stroke(0,102,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();}

  if(n>1){n--;}
  num+=sblog; // Decrease speed
  delay(100); // Regule brake
  if(num>255){num=255;} // Stop fixed

  Serial.write(num); // Speedometer
  if(num==255){screen.fill(64,64,64); screen.rect(16, 82, 10, 7);}
  ...
  if(num<=112&&num>90){screen.fill(64,64,64); screen.rect(124, 25, 10, 64);}
  }while(digitalRead(A0)==LOW&&num!=255); // End do-while structure
break;
```

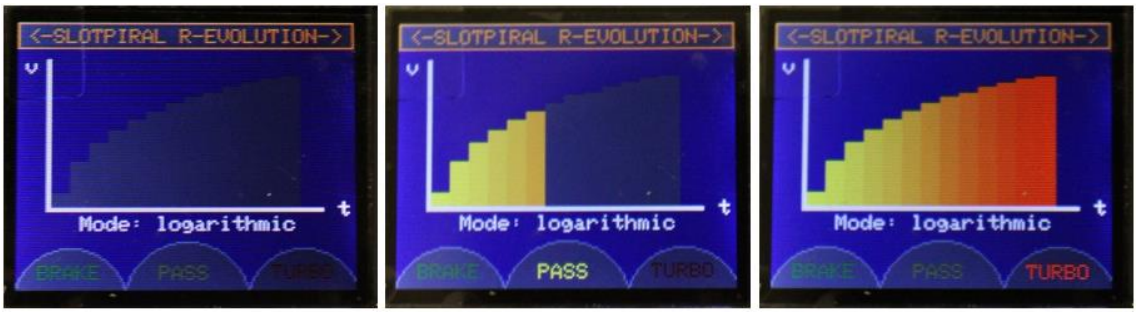

Figure 62. *Brake, pass and turbo actions screenshots in logarithmic mode.*

6.3.8 Kids mode

The third driving mode available to the slot controller is the kids mode. This mode is based on an arithmetic progression of the variable `num` for acceleration algorithm and braking, similar to the linear mode, but with a reduction of the maximum speed that the

slot vehicle can run. The following figure is a flowchart of the kids driving mode of the slot controller:

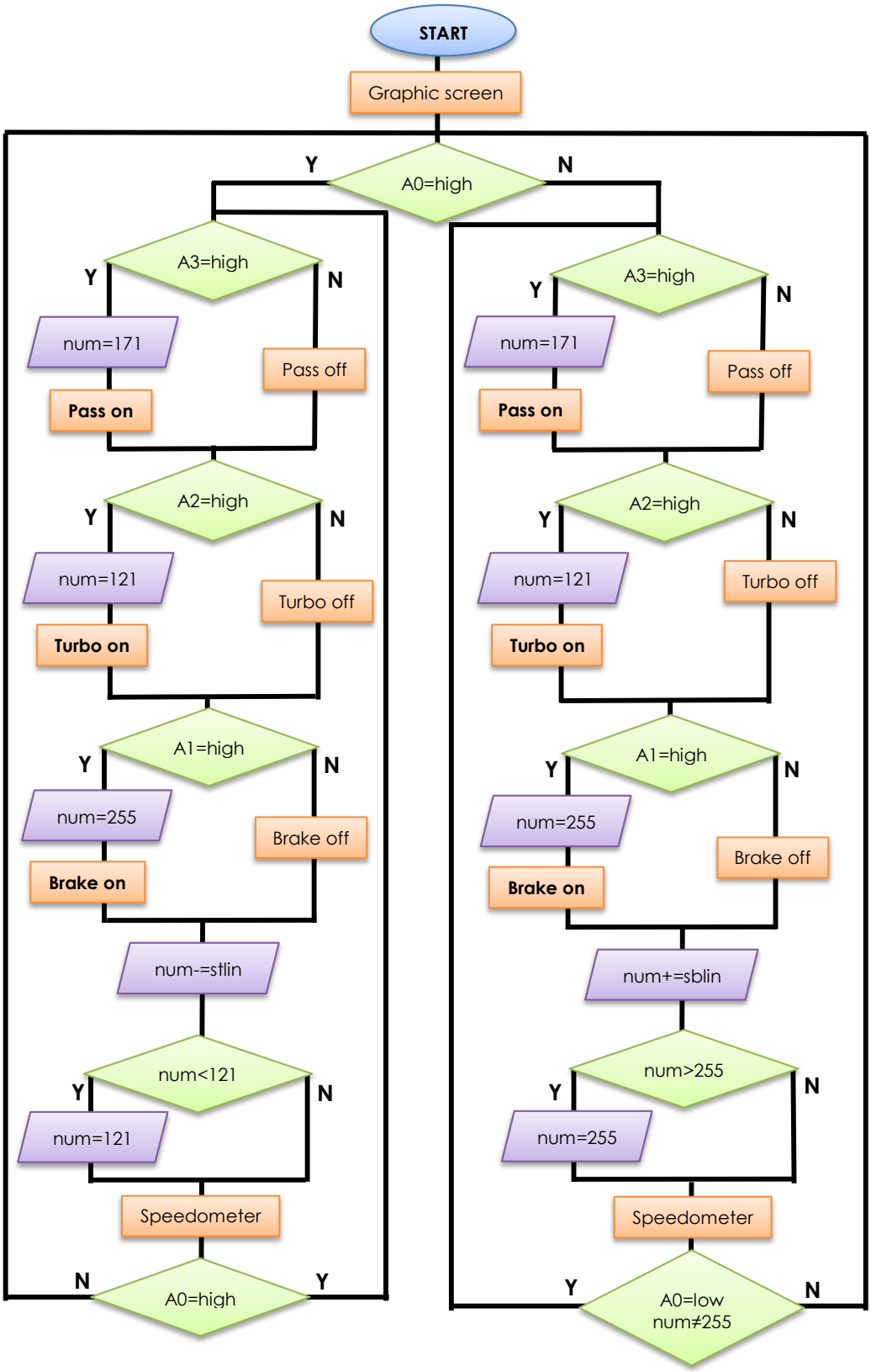

Figure 63. *Flowchart of the kids mode.*

Table 10. *Characteristics of the control variable in the kids mode.*

Characteristics of `num` variable in kids mode	
Maximum speed / turbo value	121
Zero speed value	255
Pass speed value	171
Total speed (%)	70%
Acceleration step (`stkid`)	10
Deceleration step (`sbkid`)	30

As the above Figure 63 shows, this flowchart is similar to the linear mode, but with using the reduced maximum speed that the vehicle can run.Therefore, it is an ideal driving mode for children because this speed limitation prevents the slot car run off the circuit. On the Table 10 are shown the main characteristics of the `num` control variable.

The graphical representation of this mode is also similar to the linear but with the smaller graph of velocity versus time, as shown in the following figure. The programming code of the kids mode is not discussed here because its operation principle is the same as the linear mode.

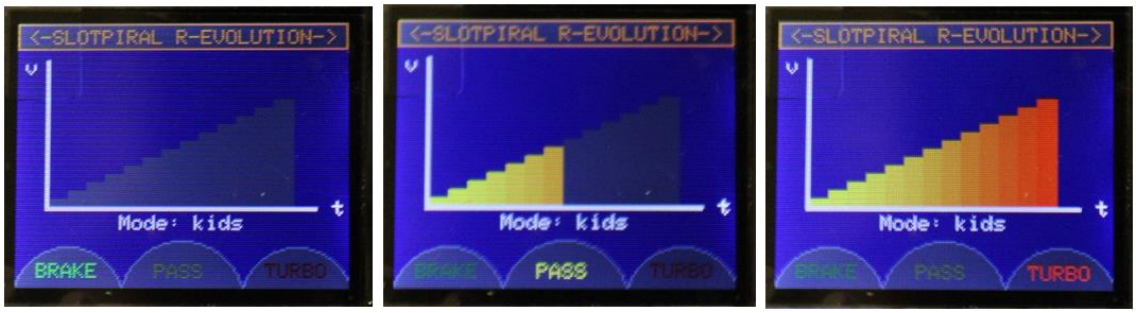

Figure 64. *Brake, pass and turbo actions screenshots in kids mode.*

6.3.9 Automatic mode

The last driving mode available to the slot controller is the automatic mode. This mode allows driving a vehicle automatically, so called “ghost car”, and it is based on an arithmetic progression of the variable `num` for acceleration algorithm and braking.

Therefore the automatic mode allows to drive a car selected by the user and to remain at a constant speed, changing lanes randomly according to an algorithm. The Figure 65 is a flowchart of the logarithmic driving mode of the slot controller for understanding how the program works.

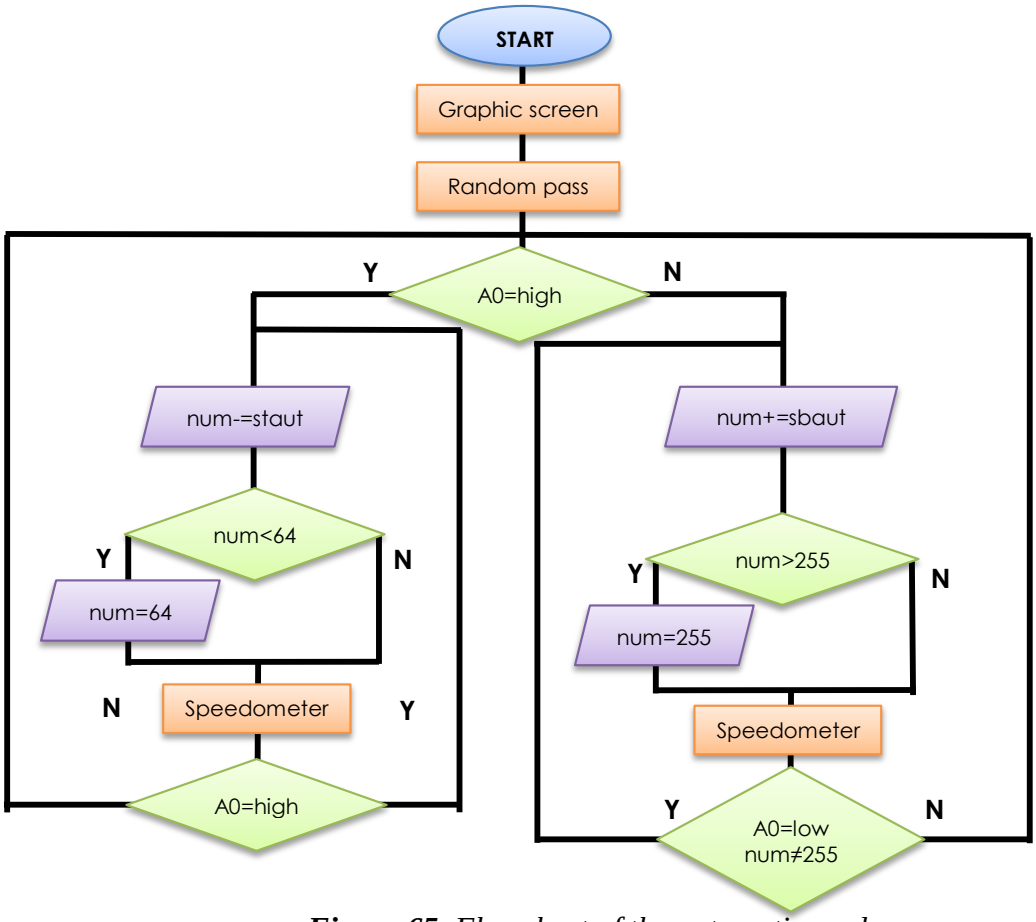

Figure 65. *Flowchart of the automatic mode.*

The program structure is divided into two main blocks. The first block is responsible for acceleration and it is composed by a `while` loop that increases the speed. The acceleration speed is established through several steps and the lane change is performed automatically by an algorithm implemented in the receiver module.

The lane change algorithm is based on an integer variable that switches among 0 and 10 randomly. Based on this value, an `if-else` structure is responsible for activating or deactivating the lane change. This algorithm was tested on a slot track to verify proper operation.

Finally, the programming code that shows the slot vehicle speed fixed on the graphic TFT LCD display.

```
int cc=random(0,10); // Random pass algorithm
  if(cc<6){digitalWrite(7,LOW);
    delay(200);
    screen.stroke(102,102,0);
    screen.text("AUTOPASS", 57, 116);
    screen.setTextSize(2);
    screen.text("<         >", 15, 112);
    screen.noStroke();
    screen.setTextSize(1);
  }else{digitalWrite(7,HIGH);
```

```
    delay(200);
    screen.stroke(255,255,0);
    screen.text("AUTOPASS", 57, 116);
    screen.setTextSize(2);
    screen.text("<         >", 15, 112);
    screen.noStroke();
    screen.setTextSize(1);}

  while(digitalRead(A0)==HIGH){ // Acceleration
    num-=staut;  // Increase speed
    delay(100); // Regule throttle
    if (num<64){num=64;} // Maximum speed fixed

    Serial.write(num); // Speedometer
    screen.stroke(255,255,255);
    if(num<=255&&num>242){screen.fill(255,255,0); screen.rect(16, 85, 10, 4);}
    ...
    if(num<=99&&num>63){screen.fill(255,51,0); screen.rect(124, 25, 10, 64);}
  } // End while structure
```

The block program corresponding to the deceleration is another `while` loop independent and it also contains algorithms to reduce the vehicle speed by fixed steps.

The programming code that shows the vehicle speed in real time on the graphic display is also run. Finally, the word `break` is used to exit from the switch statement.

```
while(digitalRead(A1)==HIGH&&num!=255){ // Deceleration
    num+=sbaut; // Decrease speed
    delay(100); // Regule throttle
    if(num>255){num=255;} // Stop fixed

    Serial.write(num); // Speedometer
    screen.stroke(255,255,255);
    if(num==255){screen.fill(64,64,64); screen.rect(16, 85, 10, 4);}
    ...
    if(num<=111&&num>63){screen.fill(64,64,64); screen.rect(124, 25, 10, 64);}
  } // End while structure
break;
```

The automatic mode is an ideal driving mode for individual enthusiasts who want to improve their driving skills alone or to test the response of a slot vehicle automatically. The operation is simple, just selecting the wished speed for the car, driven in automatic mode, and then activating the automatic lane change on the receiver module.

Figure 66. *Screenshots in automatic mode at medium speed.*

Table 11. *Characteristics of the control variable in the automatic mode.*

Characteristics of `num` variable in automatic mode	
Maximum speed / turbo value	64
Zero speed value	255
Pass speed value	171
Total speed (%)	100%
Acceleration step (`staut`)	10
Deceleration step (`sbaut`)	10
Total steps	13

The graphical representation of this mode is also similar to the linear but with the individual steps of acceleration represented in the graph, as shown in the following figure. Table 11 shows the main characteristics of the `num` control variable in the automatic driving mode.

6.4 Receiver software

In this section, the receiver module software is explained in detail, starting from the general structure of the program, local variables and libraries, also the programming code of each driving mode of the slot controller is analyzed.

The receiver is programmed to read the value of the variable `num` sent by the transmitter module and perform conversion to binary as input to the digital-to-analog converter. It also allows the lane change and activation / deactivation of the algorithm for automatic lane change mode.

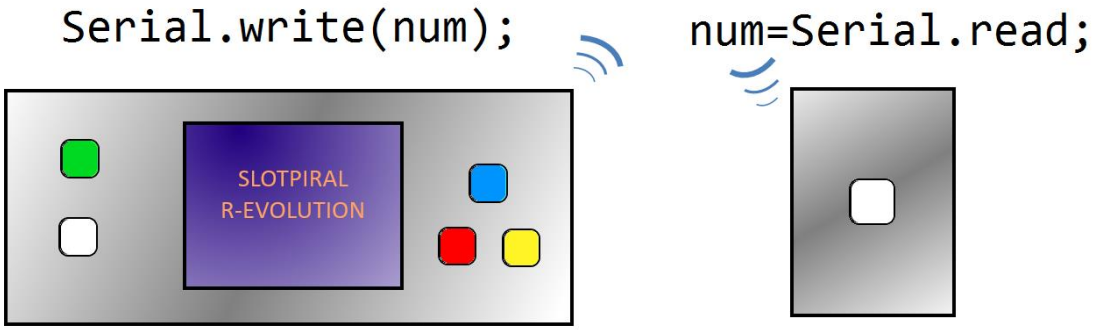

Figure 67. *Transmission and reception of the variable num.*

6.4.1 General structure

The following figure is a flowchart of the overall structure of the receiver module program.

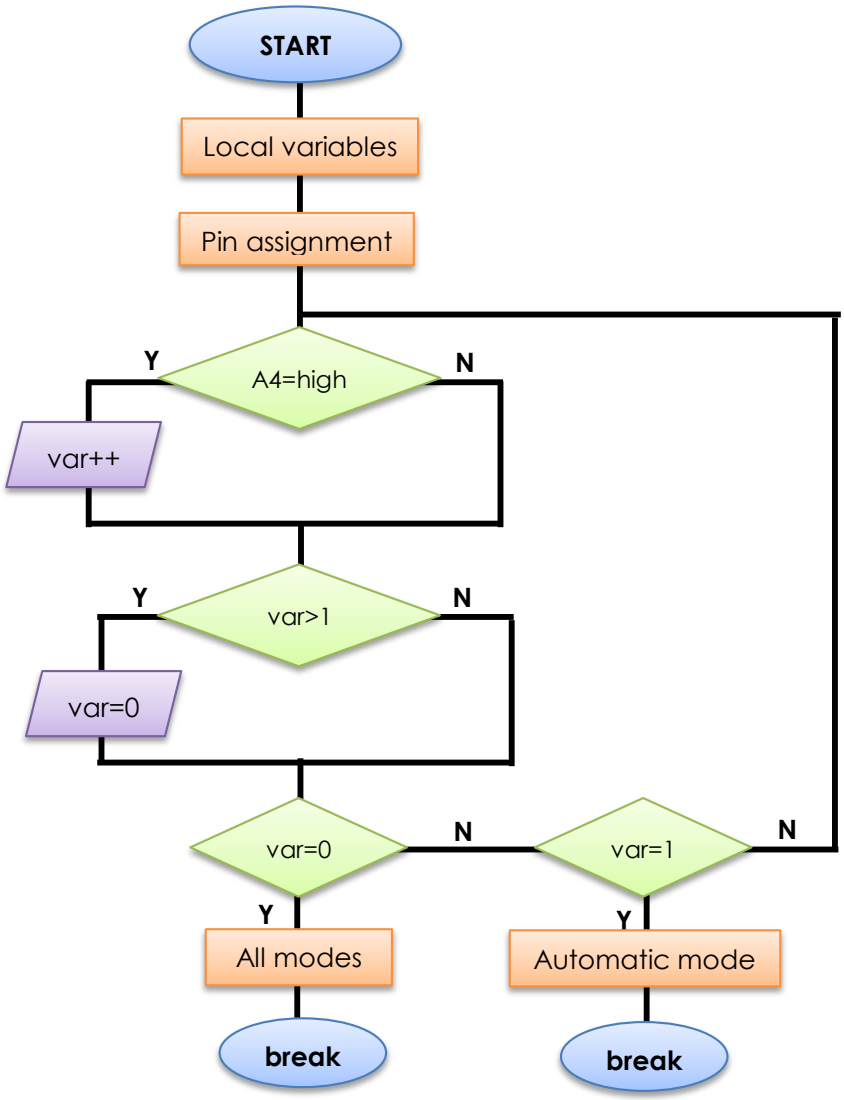

Figure 68. *General flowchart of the receiver software.*

The basic program for the previous flowchart, without detailing the program included in each `case` statement is:

```
void loop() {
if(digitalRead(A0)==HIGH){ // Mode selection
  delay(500);
  var+=1;
  if(var>1){var=0;}}
switch (var) {

  case 0: // ALL MODES
  ...
  break;
```

```
  case 1: // AUTOMATIC MODE ONLY
  ...
  break;
} // End switch structure
} // End void loop()
```

The operation of the `switch / case` statement is the same as explained for the transmitter module in previous sections.

6.4.2 Global variables

The global variables used in the program code of "Slotpiral R-evolution" receiver are shown below:

```
int num; // Counter variable
int k; // Decimal-binary conversion variable
int var; // Case structure variable
```

6.4.3 General lane change algorithm

The receiver module also has driving modes that depend on the mode used in the transmitter module. All modes use this driving mode in the receiver, except for the automatic mode. The general lane change algorithm is responsible for activating the lane change sections when the transmitter module outputs the determined decimal value to do it

The speed value calculated to allow having light lane changes within the range of values corresponds to the number 171 in decimal (base 10). The transmitter module sends a value of 171 for the variable `num` and the receiver is responsible for reading this variable, and activates the lane change when appropriate.

The code used to program the general lane change algorithm is shown below:

```
case 0: // ALL MODES_____________________________________________________________
  digitalWrite(11,LOW); // Turn off LED automatic mode
  if(Serial.available()>0){ // Receive data
    num=Serial.read(); // Receive counter variable
    Serial.print(num); // Show counter variable
    if(num!=255){digitalWrite(10,HIGH); // Tripping relay coil
    }else{digitalWrite(10, LOW);}
    if(num==171){digitalWrite(12,HIGH); // Pass enable
    }else{digitalWrite(12, LOW);} // Pass disable
  k=num; // Decimal-binary converter
  for(int i=2;i<=9;i++){
    digitalWrite(i, k%2);
    k=k/2;}
  } // End if structure
break;
```

To ensure fast braking of the slot vehicle, it is included in the code an `if-else` loop that excites a relay implemented in the transmitter module and it provides the appropriate voltage generated by the multifunction control unit to stop the slot vehicle completely.

6.4.4 Automatic lane change algorithm

The automatic driving mode has its own driving mode in the receiver module, which is explained individually. Automatic lane change algorithm is responsible for activating the lane change section automatically when according to a random algorithm implemented in the receiver module.

To work properly, you have to select the automatic mode available in the push button on the front panel of the receiver module.

The lane change algorithm is based on an integer variable that switches among 0 and 10 randomly. Based on this value, an `if-else` structure is responsible for activating or deactivating the lane change. This algorithm was tested on a slot track to verify proper operation. There is also an `if-else` structure responsible for tripping the relay coil to ensure fast braking of the slot vehicle.

The code used to program the automatic lane change algorithm is shown below:

```
case 1: // AUTOMATIC MODE ONLY_____________________________________________________
  digitalWrite(11,HIGH); // Turn on LED automatic mode
  if (Serial.available()>0){ // Receive data
    num=Serial.read(); // Receive counter variable
    Serial.print(num); // Show counter variable
    if(num==255){digitalWrite(10, LOW);
      digitalWrite(12,LOW);}
    if(num!=255){digitalWrite(10,HIGH); // Tripping relay coil
      int cc=random(0,5);
      if(cc<3){digitalWrite(12,LOW);
      delay(200);
    }else{digitalWrite(12,HIGH);
      delay(200);}}
    k=num; // Decimal-binary converter
    for(int i=2;i<=9;i++){
      digitalWrite(i, k%2);
      k=k/2;}
  } // End if
break;
```

6.4.5 Decimal to binary conversion

The speed information is transmitted numerically in a range of values ranging from 255-64 in base 10. It is necessary to convert these serial numbers to binary, for use in the 8-bit digital-to-analog converter explained in chapter 5 and designed in chapter 7.

The programming code used for this conversion is:

```
k=num; // Decimal-binary converter
for(int i=2;i<=9;i++){
  digitalWrite(i, k%2);
  k=k/2;}
```

This code divide the `num` variable by 2 and then it takes the remainder, the binary number will be the remainders read from bottom to top.

For example, to convert 22 to a binary representation:

$$22_{base\ 10} = \mathbf{0} * 2^0 + \mathbf{1} * 2^1 + \mathbf{1} * 2^2 + \mathbf{0} * 2^3 + \mathbf{1} * 2^4 = 10110_{base\ 2}$$

In binary notation the binary digits are reversed. Therefore the binary digits arise simply by iterating the process of division by 2, with remainder.

7. CONTROLLER IMPLEMENTATION

In this chapter, the implementations of the different functional blocks that are part of the slot controller are shown in detail. The evolution of circuit designs, debugging and optimization, besides the final assembly process are also shown. Furthermore, the wireless communication system comprising the transmitter module and receiver module is exposed.

7.1 Introduction

"Slotpiral R-evolution" is a controller for digital slot systems, which is based on wireless communication using radio frequency technology.

Figure 69. *Slotpiral R-evolution logo.*

To study the process of designing this command, processing is divided into two distinct parts: the transmitter module and receiver module.

7.2 Slotpiral R-evolution transmitter

The transmitter module is responsible for transmitting the actions of the push buttons and the LCD TFT screen. It also stores the control software for different driving modes. The transmitter module "Slotpiral R-evolution" slot controller includes the following functional blocks:

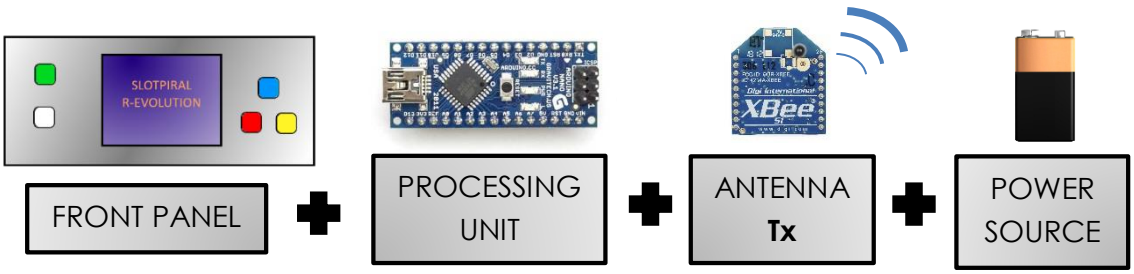

Figure 70. *Block diagram of the hardware architecture of the transmitter module.*

The transmitter module is composed of a front panel which acts as a control interface for handling the push buttons arranged on its surface, directing the information received after the activation of each push button to the next functional block architecture. It also has a LCD TFT color screen showing the different driving modes available in slot controller.

The processing unit in transmission is responsible for receiving information on the front panel and through the ATmega328 microcontroller Arduino platform different signals are processed. The microcontroller is equipped with a software that manages the operation of the slot controller, with different driving modes available.

Furthermore, the transmitter module is responsible for transmitting information in the front panel push buttons through the Tx antenna, for later analysis in the receiver module.

7.2.1 Front panel design

In the process of designing the front panel, different prototypes were made until the final implemented design.

Firstly, a controller with inverted U-shaped, inspired by video game consoles, with a LEDs bar in the center indicating the speed of the slot vehicle was designed.

Secondly, the design was refined by reorganizing the push buttons of throttle, brake and turbo to be operated with the fingers of the right hand, and pass and mode push buttons, to be operated with left hand. LEDs of the main actions were also included.

Finally, the design was modified and inspired by a steering wheel, giving to the slot controller more ergonomic. The layout of the buttons was changed and a TFT LCD color display was implemented to improve the graphical interface.

The following figure shows the evolution of the design of the front panel with the improvements discussed:

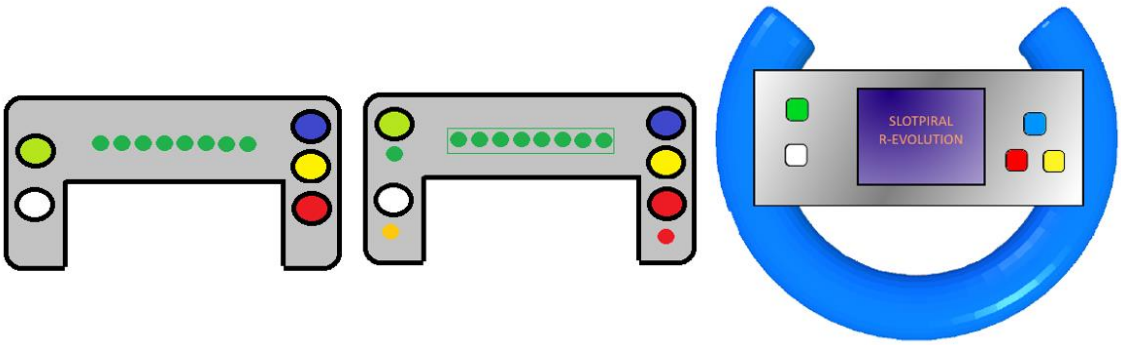

Figure 71. *Front panel evolution (left to right).*

After performing the virtual design of the front panel, the printed circuit boards were designed. The printed circuit boards of the front panel were divided into 4 modules or shields to allow a modular manufacturing and facilitate the detection and repair of breakdowns.

- **Button L shield**: it contains push buttons for pass and driving mode selection, to press it with left hand fingers.
- **Button R shield**: it contains acceleration, brake and turbo buttons, to press it with right hand fingers.
- **Central shield**: it contains the necessary connections for the TFT LCD display, power and processing unit.
- **XBee shield**: it contains the connections required by the radio frequency Tx antenna and voltage regulation.

Using the software tool OrCAD, the schematics attached in the Appendix C (p. 143) and the routing tracks of the printed circuit boards were designed (Figure 72).

Once finished the manufacturing process of the printed circuit board by chemical etching, the holes were made on the board and then the electronic components of the front panel were inserted using through-hole technology.

After cutting the remaining edges of the printed circuit board and cleaning impurities resulting from drilling and cutting process, it is necessary to continue with the next phase of the implementation of the front panel, the welding process.

As a precaution, continuity checks are carried out on the tracks of the printed circuit boards and copper remains are removed to prevent short circuits. For welding process are used flux paste, soldering wire composed of 60% tin and 40% lead, and a solder with fine tip.

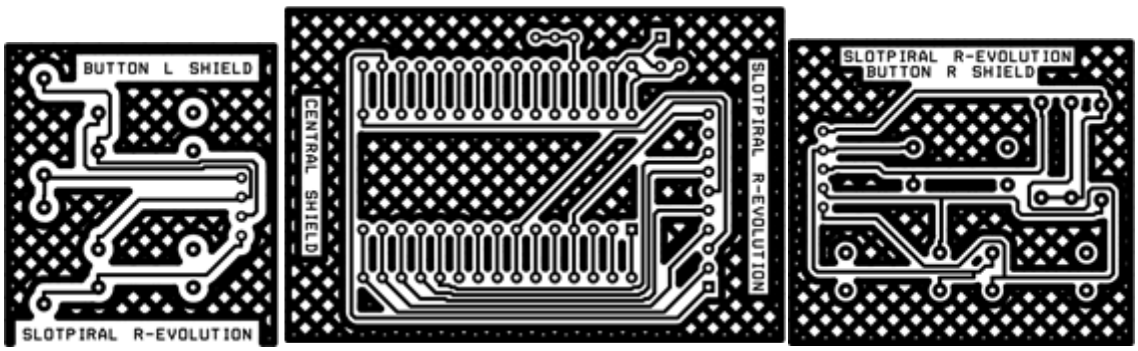

Figure 72. *Button L, central and button R printed circuit board masks.*

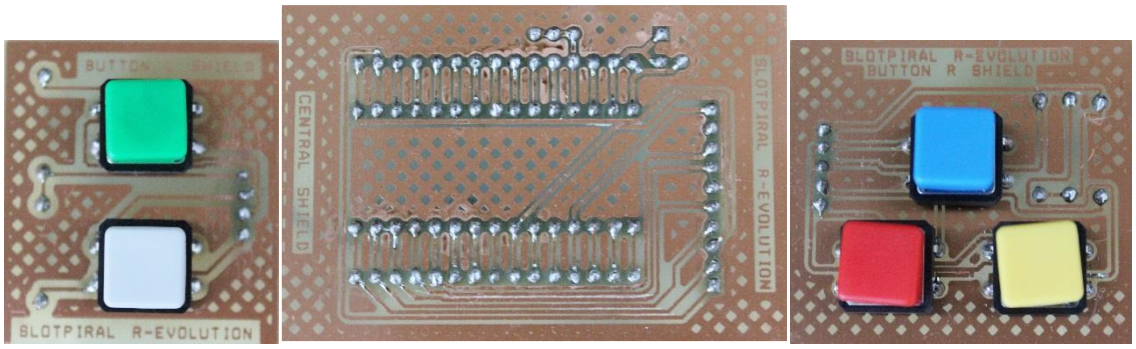

Figure 73. *Button L, central and button R printed circuit board with components.*

The minimum electronic components used to implement all shields on the front panel are the following:

- **Push buttons**: 5 push buttons are required, one for each action: throttle, braking, turbo, pass and mode selection.
- **Resistors**: 5 resistor of 10kΩ are necessary for the push buttons circuits.
- **Pin headers**: 28 pin headers spaced 2.54 mm are required to connect the different shields, 4 for the XBee shield, 4 for the button L shield, 5 for the button R shield and 15 for the central shield.
- **Pin sockets**: 40 pin sockets spaced 2.54 mm are necessary, 10 are required to connect the TFT LCD screen to the central shield and 30 are used to connect the Arduino Nano to the central shield.

Once all the welds of the PCB on the front panel are made, the conductivity is checked by a multimeter. Then, the shields are connected by cables and perfect operation is checked. Finally, after checking that it works properly, all circuits are introduced into the case previously designed.

7.2.2 Processing unit design

The processing unit is used at the open source platform Arduino, specifically the Nano model, explained in detail in chapter 5.

This platform is equipped with Atmel ATmega328 microcontroller and has the electronics necessary to connect different inputs and outputs on the pin headers arranged for this purpose on the side of the printed circuit board. One of the advantages already highlighted in the previous chapter, is that Arduino only requires a USB cable to connect to a computer and program the microcontroller so quickly.

In this section, the processing unit is designed to hardware level, indicating inputs and outputs used by the other functional blocks of the transmitter module. The main connections of the Arduino Nano platform are shown in the figure on the next page.

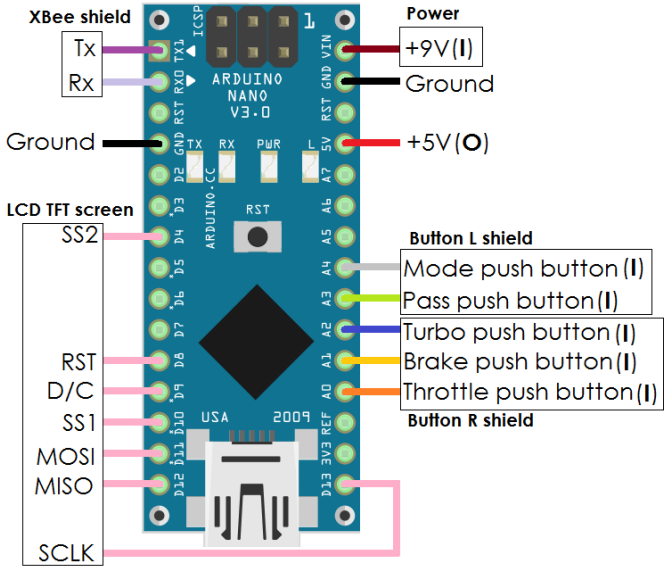

Figure 74. *Connections diagram of the transmitter processing unit.*

7.2.3 Power source design

After studying the energy needs of the transmitter module conducted in chapter 5 and the laboratory tests carried out, it is demonstrated that the power source that best suited to the project is a 9V battery. To implement power source, a switch connected in series with the 9V battery was installed, to turn on and off the slot controller easily. A battery case made of plastic was used to secure the battery in the project.

7.2.4 Communications design

To communicate the transmitter module of the slot controller, XBee shield is used. The shield is equipped with an XBee Series 1 with integrated radiofrequency antenna. This communication system uses secure point to point connection with a range up to 30 meters inside buildings, enough distance for a slot controller purpose. The actions of the front panel are processed by the Arduino Nano platform and serial data is sent by the XBee transmitter module in real time.

XBee receiver and transmitter modules were properly configured with software XCTU, linking their serial numbers to establish a secure communication [52]. The transmitter communication system is implemented by an XBee Series 1 module connected to the pin sockets of the XBee shield spaced 2 mm. Finally, the XBee shield is connected to the central shield with 4 connectors, as is shown is the Figure 75 on the next page.

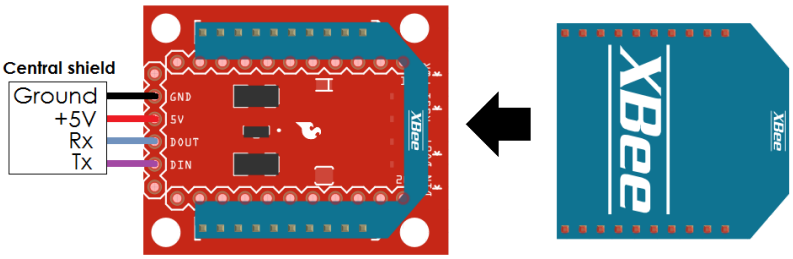

Figure 75. *Connections diagram of the XBee shield.*

7.2.5 Final implementation

In the final design stage of the transmitter module of the slot controller, the different functional blocks described in previous sections are assembled. The connections between the modules are performed using female to female wires, because all shields have headers pins.

The following picture illustrates actual result of the connections made to implement the transmitter module of the “Slotpiral R-evolution” controller:

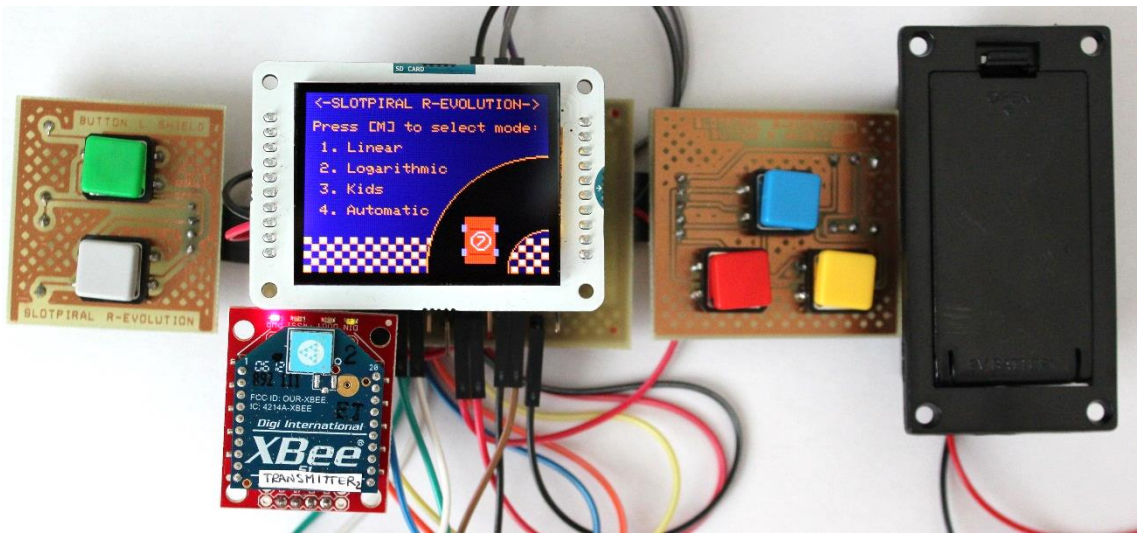

Figure 76. *Connections of Slotpiral R-evolution transmitter module.*

7.3 Slotpiral R-evolution receiver

The receiver module receives information and adapts it so it can be processed by the multifunction control unit and thus gives power the motor of the slot vehicle. The principal functional blocks of the receiver module of "Slotpiral R-evolution" are shown on the next page.

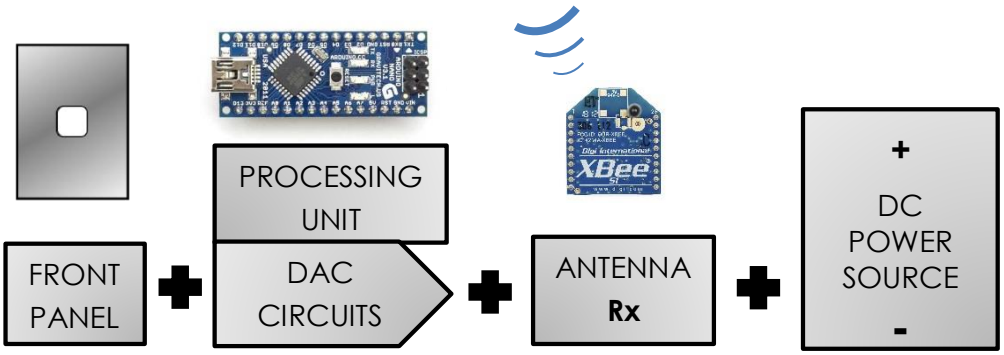

Figure 77. *Block diagram of the hardware architecture of the receiver module.*

The receiver module consists of other XBee shield and XBee Series 1 chip that are responsible for receiving data sent by the transmitter module.

After receiving the information, the receiver processing unit analyzes the data received using the software implemented in the microcontroller ATmega328 and generates a binary output with 8 bit resolution.

The output generated by the Arduino Nano platform is digital type, but must be converted to analog for the slot controller to operate properly. To convert the digital signal to analog is used a digital-to-analog converter based on R-2R resistor ladder.

The receiver module is powered by the same AC/DC transformer that powers the multifunction control unit of the slot racing system. A voltage regulator reduces the transformer output to the wished voltage to power the receiver module adequately.

7.3.1 Front panel design

The process of designing the front panel of the receiver module is simpler than the transmitter module. The design is based on a rectangular case that has a push button to select the lane change mode depending on the driving mode selected in the transmitter module.

After performing the virtual design of the front panel, the printed circuit boards were designed. The printed circuit boards of the front panel were divided into 2 modules or shields to allow a modular manufacturing and facilitate the detection and repair of breakdowns.

- **PB shield**: it contains a push button for select the type of lane change.
- **DAC shield**: it contains the necessary connections for the processing unit, power and the digital-to-analog converter.
- **XBee shield**: it contains the connections required by the radio frequency Rx antenna and voltage regulation

Using the software tool OrCAD, the schematics attached in the Appendix C (p. 143) and the routing tracks of the printed circuit boards were designed (Figure 78).

Later, the printed circuit boards were printed by chemical etching and holes were made. Finally, the electronics were welded using the same procedure outlined for implementing the front panel of the transmitter module.

The minimum electronic components used to implement all shields of the receiver front panel are:

- **Push button**: 1 push button is required.
- **Resistors**: 1 resistor of 10kΩ is necessary for the push button circuit, 7 resistor of 10kΩ and 9 resistor of 22kΩ are necessary for the DAC.
- **Pin headers**: 14 pin headers spaced 2.54 mm are required to connect the different shields, 4 for the XBee shield, 3 for the PB shield and 8 for the DAC shield.
- **Pin sockets**: 30 pin sockets spaced 2.54 mm are necessary to connect the Arduino Nano with the DAC shield. 8 individual pin sockets are required for the relay used in the DAC shield.
- **DIP package**: 1 DIP socket for 8 pins is necessary for the integrated circuit used in the DAC shield.
- **LM328**: 1 operational amplifier DIP8 encapsulated is required for DAC shield.
- **Relay**: 1 relay of 5V is necessary for the DAC shield.

Once all the welds of the PCB on the front panel are made, the conductivity is checked by a multimeter. Then, the shields are connected by cables and perfect operation is checked. Finally, after checking that it works properly, all circuits are introduced into the case previously designed.

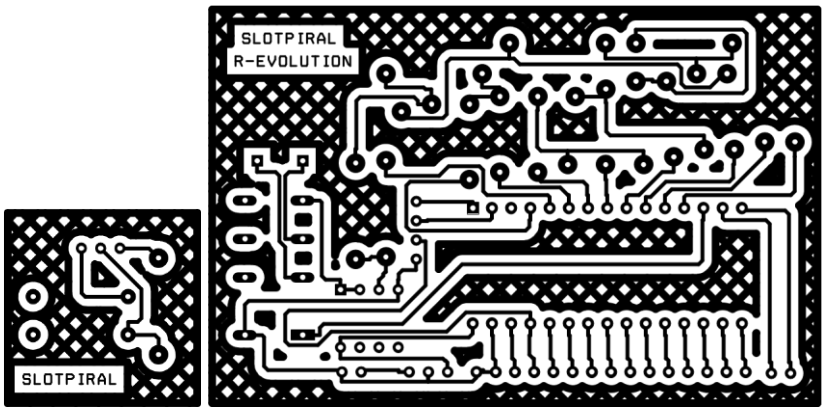

Figure 78. *PB shield (left) and DAC shield printed circuit board masks.*

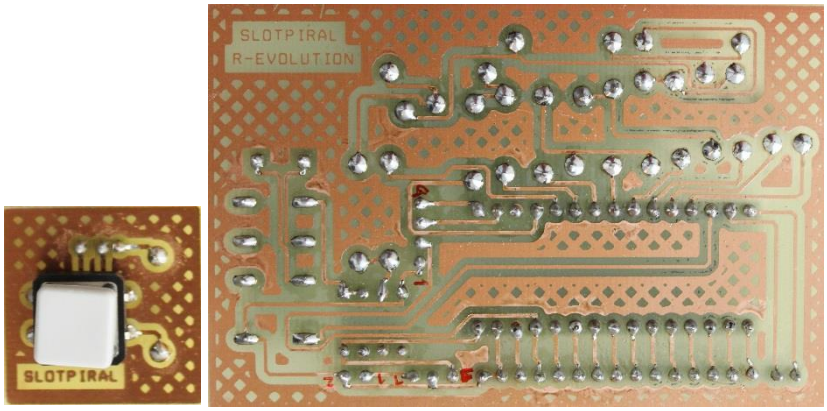

Figure 79. *PB shield (left) and DAC shield printed circuit board with components.*

7.3.2 Processing unit and digital-to analog converter design

The processing unit used for the receiver controller module is the same as the transmitting module, the Arduino Nano platform, explained in detail in chapter 5. This processing unit and the DAC are printed on the same circuit to reduce its dimensions, as shown in Figure 79.

The digital-to-analog converter is based on an R-2R resistive ladder network, as explained in detail in chapter 5, for a digital value A_{in} of a resistor ladder of 8 bits with V_{REF}=5V.

$$V_{out\ min} = A_{in} * \frac{V_{REF}}{2^8}$$

The minimum output voltage Vout min is (single step) A_{in} = 1 is:

$$V_{out\ min} = 1 * \frac{5}{256} = \mathbf{0.0195}\boldsymbol{V}$$

The maximum output voltage Vout max is (11111111) A_{in} = 255 is:

$$V_{out\ max} = 255 * \frac{5}{256} = \mathbf{4.98}\boldsymbol{V}$$

The voltage range obtained by the DAC is adequate. The minimum voltage necessary for the slot controller is 1.35V and the maximum voltage is 5.05V. With the maximum output voltage generated by the R-2R resistor ladder, the slot vehicle is completely stopped, but to ensure that 5.05V is supplied, a relay is activated and it supplies 5.05V output generated by the multifunction control unit.

The basic electrical diagram used to implement the digital-to-analog converter, with the resistive ladder and the operational amplifier is shown in the following Figure 80.

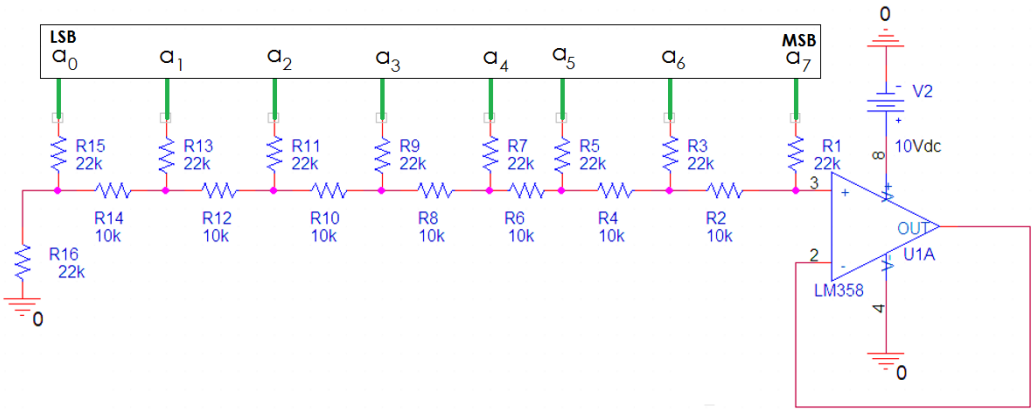

Figure 80. *Digital-to-analog converter basic electrical diagram.*

In this section, the processing unit is also designed to hardware level, indicating inputs and outputs used by the other functional blocks of the receiver module. The main connections of the Arduino Nano platform are shown in the connections diagram.

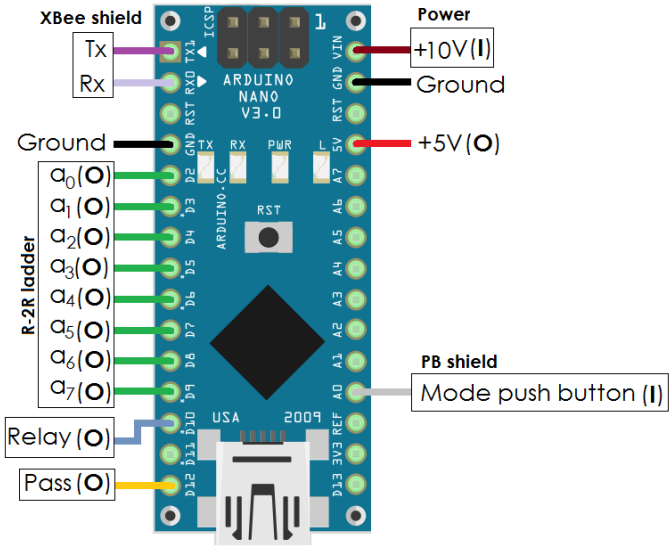

Figure 81. *Connections diagram of the receiver processing unit.*

7.3.3 Power source design

The receiver module has the peculiarity that its power is supplied via cable although the data transmission is wireless, because it is also connected to the control unit. Therefore, after studying the energy needs of the receiver module conducted in chapter 5 and the laboratory tests carried out, it demonstrated that the power source best suited to the receiver module is a 10V DC power source.

To implement this power source, the transformer that supplies the multifunction control unit is used, which generates 14V and 3A. To regulate the voltage to the proper level is

used an integrated circuit LM7810 linear voltage regulator that do not require additional components to provide 10V.

7.3.4 Communications design

The receiver module is connected to the transmitter module and the multifunction control unit by wireless and wired communications respectively.

For wireless communication, an identical XBee shield implemented in the transmitter module is used, as explained in the previous section.

Wired communication is required to send information to the multifunction control unit because it only uses 4P4C connectors to send data to the slot controllers. For wired communication unshielded twisted pair cable is used as physical media. The slot controller uses a 4P4C male connector, while the multifunction digital control unit uses a 4P4C female connector.

Implemented connections for wired communication between the receiver module and the multifunction control unit are shown in the following diagram:

- **Connector pin 1 (CN1)**: the black wire connects to normally close (NC) contactor of the relay, implemented in the DAC shield.
- **Connector pin 2 (CN2)**: the red wire connects to common (C) contactor of the relay, implemented in the DAC shield.
- **Connector pin 3 (CN3)**: the yellow wire connects to pass output pin of the processing unit, implemented in the DAC shield.
- **Connector pin 4 (CN4)**: the green wire is not connected.

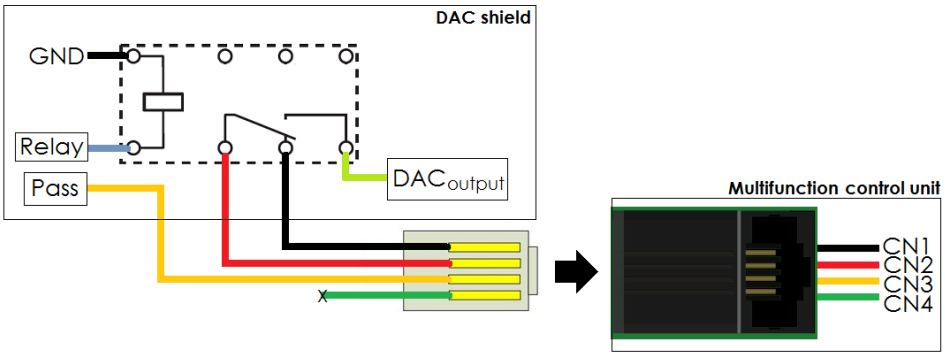

Figure 82. *Connections diagram of the wired communication.*

7.3.5 Final implementation

In the final design stage of the receiver module of the slot controller, the different functional blocks or shields described in previous sections are assembled. The connections between the modules are performed using female to female wires, because all shields have headers pins.

The following picture illustrates actual result of the connections made to implement the receiver module of the "Slotpiral R-evolution" controller:

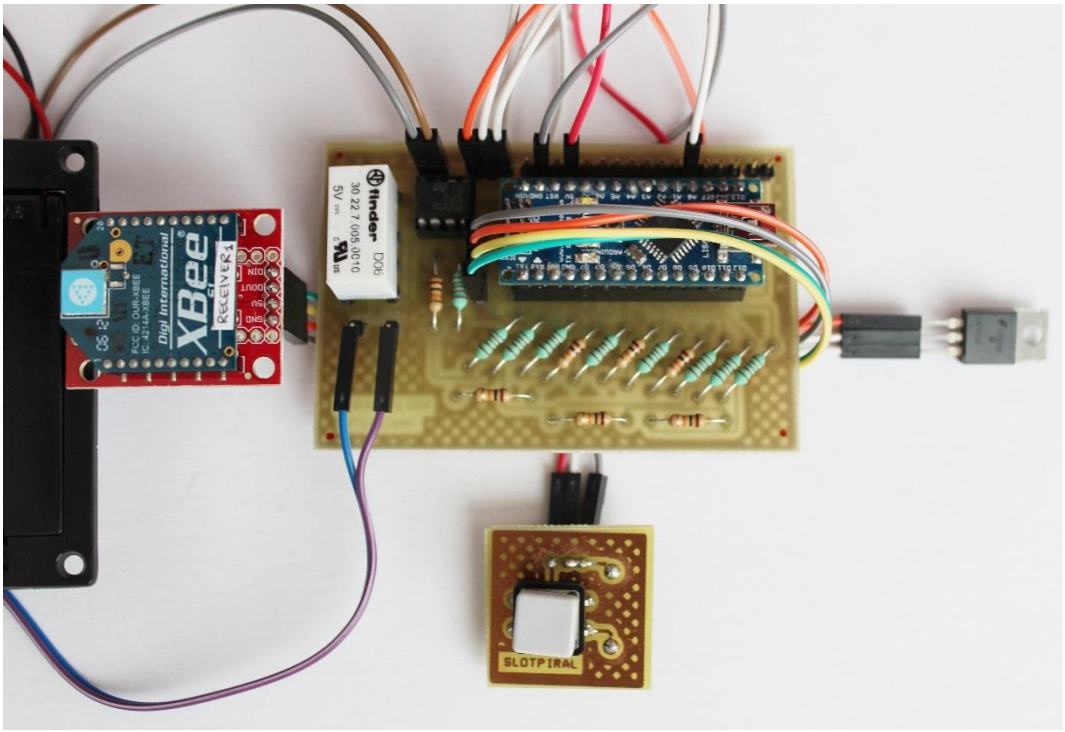

Figure 83. *Connections of Slotpiral R-evolution receiver module.*

8. ANALYSIS AND RESULTS

Once the design process, with the final implementation of the slot controller, is completed, some series of tests and experiments that prove its correct operation were performed. This chapter presents the different experiments which the slot controller was subjected to prove its proper operation. In addition, an economic study with the costs of each component used to implement the slot controller is included here. Finally, the results obtained after the development of the project are discussed.

8.1 Analysis of Slotpiral R-evolution

This section consists of an experimental analysis and relevant discussions derived from the analysis. The section is categorized into three subsections: visual analysis, laboratory tests and race analysis.

8.1.1 Visual analysis

Firstly, the slot controller was visually inspected to verify that the final result and implementation were correct. The visual analysis carried out was the following:

- **Push buttons**: all the push buttons of the transmitter module were visually checked to ensure that its disposal was accurate and responsive when pressed. The individual push button on the receiver module was also tested.
- **TFT LCD screen**: the color display was analyzed to verify that the brightness calibration was enough to operate with different types of lighting, without presenting reflections affecting their accurate use.
- **Software**: the program implemented in the transmitter module was simulated to verify that all driving modes were run properly.

8.1.2 Laboratory tests

Secondly, several laboratory tests were conducted to demonstrate that slot controller operation is adapted to the standard theoretical assumptions.

Using an oscilloscope, the output voltage signal generated by the receiver module into the connector CN2 was sampled. Also it was verified that the voltage range generated was correct. The connector CN3 was also sampled to verify the changed lane action.

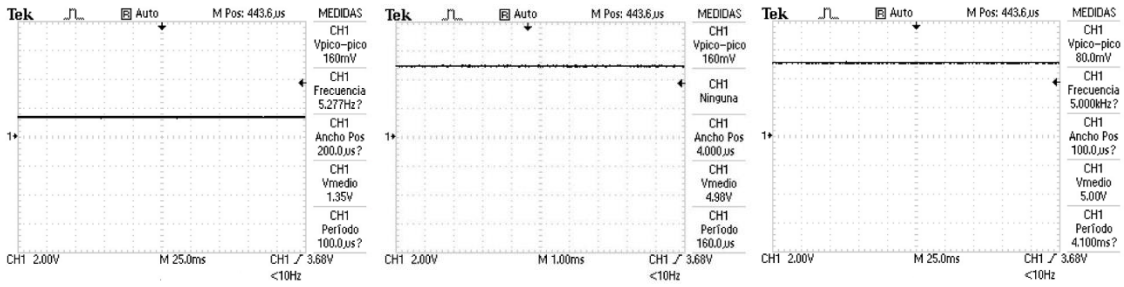

Figure 84. *Oscilloscope screenshots of the laboratory tests.*

The actions implemented in each driving mode were sampled by the oscilloscope. The Figure 84 shows screenshots of the main actions of the slot controller operating in the linear mode are shown:

- **Maximum speed / turbo**: the suitable maximum speed for the slot car engine is equivalent to 1.35V, as explained in chapter 4.
- **Minimum speed**: the value equivalent to the minimum speed voltage is calculated in chapter 7 and corresponds to the maximum value of voltage generated by the digital-to-analog converter, it is 4.98V.
- **Total brake**: to ensure that the vehicle's engine stops completely, 5.05V are obtained from the multifunction control unit as explained in chapter 7.
- **Lane change / pass**: to activate the lane change is generated a pulse of 5V.

8.1.3 Race analysis

Finally, after the positive results of previous experiments, the driver was sent to Spain. In particular, the definitive experiments were carried out in the laboratories of the Technical University of Cartagena (UPCT), using the digital slot car system manufactured by the Spanish company *Ninco*.

The practical experiments are focused on three ways:

- The response of different driving modes in the circuit was analyzed. This analysis was done by professional hobbyist, to know their point of view as experts who cataloged driving modes of the slot controller as very realistic.
- The transmission and reception speed of the modules was checked, resulting in real-time communication without perceptible delay.
- Also were performed tests to measure the maximum coverage area of the slot controller, overcoming the physical limits of the building where the laboratory was located, more than the adequate result.

In conclusion, after all experiments it was demonstrated that the slot controller "Slotpiral R-evolution" worked properly, satisfying all the requirements previously supposed.

Figure 85. *UPCT experimental digital slot car system.*

8.2 Cost report

In this section is presented the cost report of the project developed, showing separately the cost of the transmitter module and receiver module. In the summary of costs, the individual prices of each of the electronic components used are tabulated and the total price amounting development is calculated.

8.2.1 Transmitter module cost report

The cost report of the material used for the implementation of the transmitter module of the digital slot controller "Slotpiral R-evolution" appears tabulated below:

Table 12. *Transmitter module cost report.*

Description	Unit price	Quantity	Cost
Resistor 10kΩ, 0.25 W	0.03	5	0.15
Colour push button	0.31	5	1.55
Arduino TFT LCD screen	19	1	19
Arduino Nano V3.0 compatible	7	1	7
XBee Series 1	24	1	24
XBee Explorer Shield	8.85	1	8.85
Copper clad board	1.9	1	1.9
Pin headers 2.54mm	0.78	1	0.78
Pin sockets 2.54mm	0.79	1	0.79
Switch	0.95	1	0.95
9V Battery + case holder	3.5	1	3.5
Female to female wires	1.7	1	1.7
Plastic case	7.58	1	7.58
TOTAL			**77.75 €**

8.2.2 Receiver module cost report

The cost report of the material used for the implementation of the receiver module of the digital slot controller "Slotpiral R-evolution" appears tabulated below:

Table 13. *Receiver module cost report.*

Description	Unit price	Quantity	Cost
Resistor 10kΩ, 0.25 W	0.03	8	0.24
Resistor 22kΩ, 0.25 W	0.03	9	0.27
Colour push button	0.31	1	0.31
Arduino Nano V3.0 compatible	7	1	7
XBee Series 1	24	1	24
XBee Explorer Shield	8.85	1	8.85
Copper clad board	1.9	1	1.9
Pin headers 2.54mm	0.78	1	0.78
Pin sockets 2.54mm	0.79	1	0.79
Switch	0.95	1	0.95
LM358	0.29	1	0.29
4P4C connector	0.14	1	0.14
2.1mm barrel jack	0.5	1	0.5
Female to female wires	1.7	1	1.7
Plastic case	4.58	1	4.58
TOTAL			**52.3 €**

8.2.3 Total cost report

The cost report of the material used for the implementation of the transmitter and receiver module of the digital slot controller "Slotpiral R-evolution" appears tabulated below:

Table 14. *Total cost report.*

Description	Cost
Transmitter module	77.75
Receiver module	52.3
TOTAL	**130.05 €**

8.3 Discussion of the results

The results of this Master's Thesis are the design, development and implementation of an innovative wireless electronic controller for digital slot car applications that allows more precise control of slot vehicles as well as new features that make it more intuitive. The slot controller is called "Slotpiral R-evolution" because it resolves the current problems of commercial slot car controllers and it provides new functionalities to the slot car racing.

The final results are divided into two main fields: one dedicated to the development of electronic hardware and another focused on the development of the software control interface.

The hardware field was accomplished using an Arduino Nano platform that combines a low cost and high performance Atmel 8-bit AVR RISC-based microcontroller, with all the conditioning electronics to obtain an accurate control of the slot vehicles parameters. Throttle, brake, pass and turbo actions were first implemented in a slot controller with push buttons and the results obtained were really successful. The TFT LCD color allows the ease of use and the wireless communication gives more independence.

The software field developed the slot controller program and it is responsible for provide 4 different driving modes (linear, logarithmic, kids and automatic). This driving modes are perfectly suited to the individual characteristics of the hobbyists and the conditions of each slot racing circuit. The graphic interface designed for the slot controller screen is very intuitive, allowing to be used by children and adults. Wireless communication between the transmitter and receiver module of the slot controller is safe and it is set automatically through the program implemented.

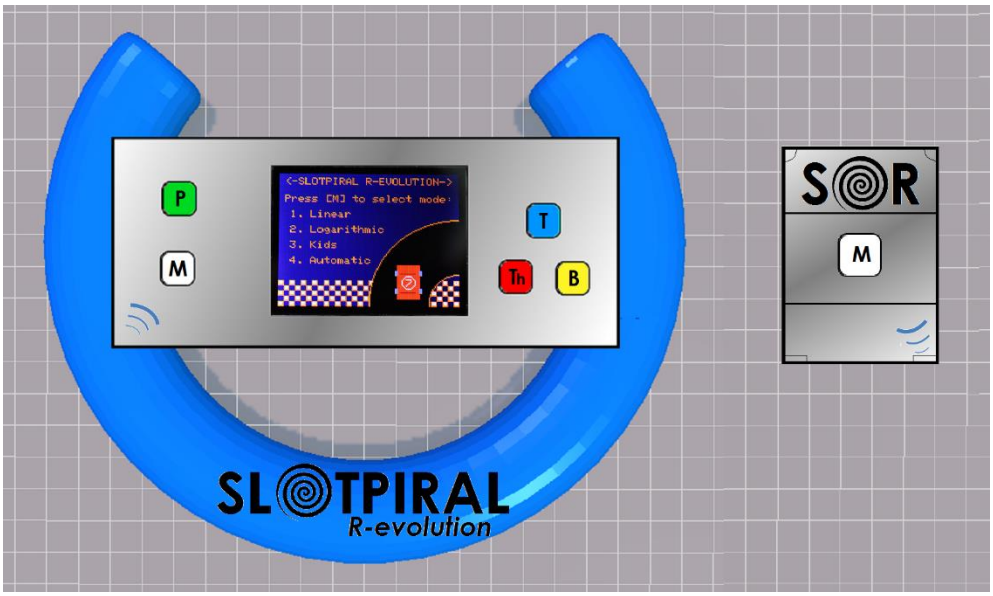

Figure 86. *3D designs of Slotpiral R-evolution transmitter (left) and receiver.*

The industrial design of the external appearance of the product was also studied, a case design proposal for the slot controller was made to implement by 3D printer (Figure 86). For the transmitter module, the case design is inspired by a steering wheel with ergonomic pads on the sides of the wheel to operate the slot controller with both hands. For the receiver module, the case design is inspired by car top view with a push button over the roof.

Throughout the process of realization of this thesis, the results achieved have been demonstrating in detail in all previous chapters. It was necessary to put constant feedback on product designed to ensure that at the end of implementation, the final results are completely effective.

As conclusion, the results have been satisfactory to achieve the baseline tasks. The “Slotpiral R-evolution” controller has a color display that provides an intuitive interface with the placement of the push buttons. The wireless communication allows more independence and different driving modes that provide new features never seen in previous slot car systems. Probably, the most important merit achieved is the development of a real project that has interest to people.

9. CONCLUSIONS

In this last chapter, the conclusions reached at the end of the project are set out and the most relevant problems encountered during the implementation thereof are cited. Finally, there is also a section with potential future works related to research line developed in this project are proposed.

9.1 Final conclusion

The slot car racing or slot racing has been a popular form of entertainment which has brought children and adults together since its inception, due to the simplicity of operation which makes it suitable for all age ranges. The slot racing had its heyday a few decades ago, but with the emergence of new video game consoles and the rise of personal computers, it lost the privileged position that had held for years. But today still enjoys popularity; there are manufacturers of more genuine slot circuits, the vehicles are adapted to the latest models created in the market and there are many people interested in acquiring the most recent developments in this field.

Although there is currently technology market quite evolved, the advances and innovations made by the major manufacturers of slot car systems are very scarce. As a general trend, manufacturers are more focused on improving the visual appearance of the vehicles and create new models, instead of in improve the control system used for these. Slot controllers are a field with few improvements, just highlights the wireless system implemented by some manufacturers. All controllers based its operating principle in the traditional format of handle with throttle trigger. Therefore this Master's Thesis appears to shed light to the world of Slot, with “Slotpiral R-evolution” have been corrected the deficiencies in innovation of the main slot manufacturers, providing an evolved command and fitted with greater possibilities for the control of the slot cars.

“Slotpiral R-evolution” has new features such as custom push buttons for each action and a TFT LCD color screen as a visual interface that makes driving slot vehicles easier. The display shows the different driving modes available to the user, the activities carried out in real time and also has a speedometer that gives greater realism to the slot controller. One of the basic priorities of this controller is to make the slot racing hobby accessible to all kinds of people, providing a more intuitive and precise control, as opposed to other controllers which are created only for the professional hobbyist sector. Also emphasize the development of wireless communication system, which allows an improvement in the dynamics of slot racing, giving freedom to the users to move around

the circuit without losing visibility and with a range limited only by the own visual capabilities of each person.

For the realization of this thesis was necessary to combine knowledge from several fields gained during more than 5 years of higher education at the Technical University of Cartagena (Spain), as well as expanding technological background to overcome the challenge posed by this ambitious project. A thorough study of the world of slot racing was conducted by analyzing the different slot car systems used today, getting feedback through discussions with professional and novice hobbyists about the shortcomings of commercial slot controllers and the features that they would implement in its own controllers.

Even though the results obtained after the completion of the project may reflect the difficulty of the diverse research process carried out, possibly not approaching to show the real effort throughout the development of this thesis. There were numerous problems and setbacks encountered along the way, but with perseverance and motivation were resolved to achieve the goal set from the beginning.

The practical verification of the operation of the slot controller was an important stimulus, although there is no doubt the weight of the theoretical part of the project, in practice does not always correspond with the theoretical framework, so the fact that the system works and can be verified in reality with satisfactory performance, is a significant value when analyzing the results.

As a final conclusion, let me emphasize that the acquisition of knowledge related to the project has been developed remarkable. Hence, the success of the results and the wonderful experience lived during the months of project development at the Tampere University of Technology (Finland), make it attractive to know that this Master's Thesis can be the source of other projects related to the digital slot racing, thus justifying the enormous satisfaction that the author has. With "Slotpiral R-evolution" a course is finished, but a new research field starts... because research is the key to the future.

9.2 Future works

There are various future projects that can be performed to improve the slot car racing, because this hobby is an open field to the imagination and creativity of hobbyists.

This Master's Thesis has been the seed of a future project that will be presented in the Technical University of Cartagena as part of a Master's Thesis entitled *"Slotpiral R-evolution Android: wireless electronic controller for digital slot car applications. Hardware and software design of the product"*.

This project continues the research field open with "Slotpiral R-evolution", to implement its features presented in a new format. The slot controller stops to be a physical

object to go beyond and become an application for the Android operating system. A new receiver module will be implemented to enable smartphones and tablets driving a slot vehicle in an innovative way and enjoy the slot car racing without slot controllers.

REFERENCES

[1] R. W. Reed, Know About Model Roadracing, Editors and Engineers Ltd., 1966, 96 p. ASIN: B000RDXPL6.

[2] R. Cullen, Slot Car Patent by Albert E. Cullen. Available: http://www.lightningspeedway.com/images/pat3.gif

[3] R.F. Dempewolff, Table-Top Car Racing, Allen & Unwin, Apr 1965, 155 p. ISBN-13: 978-0047960062.

[4] D.J. Laidlaw-Dickson, Table Top Rail Racing Track & Cars, Model Maker 4, Dec 1954.

[5] V. N. Sinclair, A Pioneer Electric Rail Track, Model Maker 7, Feb 1957.

[6] D. J. Laidlaw-Dickson, Scalex Goes Electric, Model Maker 7, Apr 1957.

[7] Wikimedia Commons, "Scalextric-Tin-Cars-restored.png", 2008. Available: http://commons.wikimedia.org/wiki/File:Scalextric-Tin-Cars-restored.png

[8] D. J. Laidlaw-Dickson, Slot-Racing De Luxe, Model Maker 7, Nov 1957.

[9] D. J. Laidlaw-Dickson, Introducing VIP, Model Maker 7, Oct 1957.

[10] T. Cook, Tyco Twin Turbo Trains, No.7438, Tony Cook's HO-Scale Trains Resource.

[11] J. Alonso, R. Pérez, J. Torre, Scalextric: Historia y nostalgia, Almuzara, Nov 2008. ISBN-13: 978-8496968455.

[12] Circuito Slot, "scalextric1967.jpg". Available: http://www.circuitoslot.es/coches-exin/scalextric1967.jpg

[13] A. Norman, R. Gillham, Scalextric Past and Present: The Ultimate Guide, ADH Publishing Ltd, Jan 2015. 700 p. ISBN-13: 978-0955449918.

[14] J. Muraday, Ninco 1993 – 2003, 10º Aniversario, 2003.

[15] J. Lindfors, Sähköautoradalla kisa on totista mutta vauhti vaaratonta, Apr 2014. Available: http://yle.fi/aihe/artikkeli/2014/04/09/sahkoautoradalla-kisa-totista-mutta-vauhti-vaaratonta#media=102455

[16] I. Keltanen, M, Paalosmaa, Film poste 131. Electrico auto track, Nov 1968.

[17] Wikimedia Commons, "SlotcarElecCircuit.png", Feb 2007. Available: http://en.wikipedia.org/wiki/File:SlotcarElecCircuit.png

[18] D. Chang, Digital Slot Car Racing in 1/32 Scale: Covering: Scalextric, Carrera, Ninco, SCX and Specialist Digital Systems, The Crowood Press Ltd., Nov 2011, 208 p. ISBN-13: 978-1847973061.

[19] R. Gillham, Scalextric: Cars and Equipment, Past and Present, Haynes Manuals Inc, Aug 1998, 208 p. ISBN-13: 978-1859604328.

[20] Haydon Switch & Instrument Inc., Designer's Corner: useful technology for your idea file, Global Design News, 2001.

[21] L.H. Hertz, The Complete Book of Model Raceways and Roadways, Temple Press Books, 1965, 216 p. ASIN: B0000CMJ0L.

[22] Aurora Plastics Corporation, The Complete Handbook of Model Car Racing, Prentice Hall, Jun 1967. ISBN-13: 978-0131610002.

[23] Material edit from Wikimedia Commons, “Slotcar-Motors-pt1.gif”, Aug 2007. License: CC BY-SA 3.0. Original available: http://en.wikipedia.org/wiki/File:Slotcar-Motors-pt1.gif

[24] Material edit from Wikimedia Commons, “Slotcar-motors-pt2.gif”, Aug 2007. License: CC BY-SA 3.0. Original available: http://en.wikipedia.org/wiki/File:Slotcar-motors-pt2.gif

[25] Wikimedia Commons, “Slotcar scale comparison.png”, Apr 2007. Available: http://en.wikipedia.org/wiki/File:Slotcar_scale_comparison.png

[26] T. Graham, Greenberg's Guide to Aurora Slot Cars, Greenberg Pub, Aug 1995. ISBN-13: 978-0897784009.

[27] G. Preston, Race Aurora A.F.X., Special Interest Model Books, Oct 1982, 100 p. ISBN-13: 978-0852427279.

[28] D. Chang, The Slot Car Handbook: The Definitive Guide to Setting-Up and Running Scalextric Style 1/32 Scale Ready-to-Race Slot Cars, The Crowood Press Lt,. Apr 2007, 128 p. ISBN-13: 978-1861269164.

[29] Wikimedia Commons, “Three lane slot car track.jpg”, Oct 2007. Available: http://commons.wikimedia.org/wiki/File:Three_lane_slot_car_track.jpg

[30] Material edit from P. Nikrus, Slot Car Tracks 3, Nov 2011. License: CC BY-NC-SA 2.0. Original available: https://www.flickr.com/photos/skrubu/6430097277/in/photolist-aNcVYV-aNcVRV-aNcVWt-aNcVUa

[31] R. W. Greenslade, History of Electric Model Roads and Racetracks, 1908-85: A Life of Collecting and Information on Model Racetracks, Leisure-time Publications Ltd, Jan 1985, 164 p. ISBN-13: 978-0948793004.

[32] Wikimedia Commons, "Slot-Controllers.png", Sept 2007. Available: http://en.wikipedia.org/wiki/File:Slot-Controllers.png

[33] R. Gillham, Scalextric: The Ultimate Guide, J H Haynes & Co Ltd., Nov 2008, 360 p. ISBN-13: 978-1844255368.

[34] Scalextric 2014 – 2015 catalogue. Available: http://scalextric.es/assets/files/CAT.SCALEXTRIC-2014_baja_.pdf

[35] Carrera 2015 catalogue. Available: http://www.carrera-toys.com/fileadmin/user_upload/downloads/Carrera_Katalog_2015_US_DS_Ansicht_final.pdf

[36] SCP-2 manual. Available: http://slot.it/Download/SCP1_Manuals/manuale-multi-2.0.pdf

[37] Scorpius Wireless Official website. Available: http://www.scorpiuswireless.com/wp-content/uploads/2015/02/scorpius-controller.png

[38] S. Ros Navarro, Desarrollo de Mando Electrónico para Slot Digital. Diseño de Prototipo Electrónico, Universidad Politécnica de Cartagena, 170 p. Sep 2012.

[39] Ninco 2015catalogue. Available: http://issuu.com/nincohobby/docs/catalogue_2015_eng_final_web_feda86839fd05b

[40] Ninco Digital System Instructions. Available: http://www.modelrectifier.com/resources/ninco/N-Digital%20%28Set%29%20Instruction%20Manual.pdf

[41] J. A. Martínez Jiménez, Desarrollo de Mando Electrónico para Slot Digital. Diseño de Interfaz de Control, Universidad Politécnica de Cartagena, 176 p. Sep 2012.

[42] N. Zambetti. "Arduino Serial board". Available: http://www.arduino.cc/en/uploads/Main/arduino.jpg

[43] Arduino Nano 3.0 schematic. Available: http://arduino.cc/en/uploads/Main/ArduinoNano30Schematic.pdf

[44] C. Amariei, Arduino Development Cookbook, Packt Publishing, Apr 2015, 246 p. ISBN-13: 978-1783982943.

[45] G. Radulov, P. Quinn, H. Heght, Smart and Flexible Digital-To-Analog Converters, Springer, Jan 2011, 324 p. ISBN-13: 978-9400703469.

[46] Material edit from Innolux. Original available: http://www.innolux.com/Files/Images/Technology/tft_lcd-1.gif

[47] Material edit from Arduino. Original available: http://arduino.cc/en/uploads/Main/GLCDFront_450.jpg

[48] Duracell 6LR61 battery datasheet. Available: http://ww2.duracell.com/media/en-US/pdf/gtcl/Product_Data_Sheet/NA_DATASHEETS/MN1604_6LR61_US_CT.pdf

[49] LM7810 Fairchild Semiconductor datasheet. Available: https://www.fairchildsemi.com/datasheets/LM/LM7810.pdf

[50] IEEE 802.15 Standars. Available: http://standards.ieee.org/about/get/802/802.15.html

[51] XBee Series 1 datasheet. Available: http://www.digi.com/pdf/ds_xbeemultipointmodules.pdf

[52] R. Faludi. Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing, O'Reilly Media, Jan 2011. 322 p. ISBN-13: 978-0596807733.

[53] Gravitech, Arduino Nano 3.0 User Manual. Available: http://site.gravitech.us/Arduino/NANO30/Arduino_Nano3_0.pdf

Note[1]: All images that are not referenced in this list are courtesy of the author of this Master's Thesis. The photographs were taken with a camera Canon 600D and are licensed under a *Creative Commons Attribution - NonCommercial 4.0 International* License (CC BY-NC 4.0).

Note[2]: All diagrams and schematics that are not referenced in this list are courtesy of the author of this Master's Thesis. The diagrams were designed with open source software GIMP 2.8.14 and Fritzing 0.9.2b. The elecronic schematics were designed with OrCAD Family Release 9.2 Lite Edition.

APPENDIX A: Source code of Slotpiral R-evolution transmitter

```
//TRANSMITTER SLOTPIRAL R-EVOLUTION PROGRAM CODE
//GLOBAL VARIABLES
int stlin=14; // Throttle steps linear mode
int sblin=38; // Brake steps linear mode
int stlog=10; // Throttle steps logarithmic mode
int sblog=38; // Brake steps logarithmic mode
float n=1; // Formula steps logarithmic mode
int stkid=10; // Throttle steps kids mode
int sbkid=30; // Brake steps kids mode
int staut=10; // Throttle steps automatic mode
int sbaut=10; // Brake steps automatic mode
int num=255; // Counter initialize min speed
int var=0; // Switch case structure variable
int ctlog=147; // Logarithmic mode constant
int cte=147; // Mode logarithmic constant
int pass=171; // Pass constant
int a=0; // Graphical control variables
int b=0;
int c=0;
//LIBRARIES
#include <SPI.h> // SPI communication library
#include <TFT.h> // Arduino TFT library
#define cs   10 // Define screen CS pin
#define dc   9 // Define screen DC pin
#define rst  8 // Define screen RESET pin
TFT screen = TFT(cs, dc, rst);

void setup() {
Serial.begin(9600); // Initialize serial communication
pinMode(A0,INPUT); // Throttle push button (Th)
pinMode(A1,INPUT); // Brake push button (B)
pinMode(A2,INPUT); // Turbo push button (T)
pinMode(A3,INPUT); // Pass push button (P)
pinMode(A4,INPUT); // Mode push button (M)
screen.begin();  // Initialize the screen
// WELCOME MESSAGE
screen.background(0,0,0);  // Background colour
screen.stroke(255,128,0); // Text colour
screen.text("<-SLOTPIRAL R-EVOLUTION->", 5, 3);
screen.fill(255,255,255); // Fill colour
for(int i=0; i<13; i++){screen.rect(0, 123-10*i, 5, 5);}
for(int i=0; i<12; i++){screen.rect(155, 118-10*i, 5, 5);}
delay(200);
screen.rect(0, 123, 5, 5);
delay(200);
screen.rect(150, 123, 5, 5);
```

```
delay(200);
screen.rect(5, 118, 5, 5);
delay(200);
screen.rect(145, 118, 5, 5);
delay(200);
screen.rect(10, 123, 5, 5);
delay(200);
screen.rect(140, 123, 5, 5);
delay(200);
screen.text("CREDITS:", 55, 20);
screen.rect(15, 118, 5, 5);
delay(200);
screen.rect(135, 118, 5, 5);
delay(200);
screen.rect(20, 123, 5, 5);
delay(200);
screen.rect(130, 123, 5, 5);
delay(200);
screen.rect(25, 118, 5, 5);
delay(200);
screen.rect(125, 118, 5, 5);
delay(200);
screen.text("Santiago Ros Navarro", 20, 40);
screen.rect(30, 123, 5, 5);
delay(200);
screen.rect(120, 123, 5, 5);
delay(200);
screen.rect(35, 118, 5, 5);
delay(200);
screen.rect(115, 118, 5, 5);
delay(200);
screen.rect(40, 123, 5, 5);
delay(200);
screen.rect(110, 123, 5, 5);
delay(200);
screen.text("Technical University of", 10, 60);
screen.text("Cartagena (UPCT)", 35, 70);
screen.rect(45, 118, 5, 5);
delay(200);
screen.rect(105, 118, 5, 5);
delay(200);
screen.rect(50, 123, 5, 5);
delay(200);
screen.rect(100, 123, 5, 5);
delay(200);
screen.rect(55, 118, 5, 5);
delay(200);
screen.rect(95, 118, 5, 5);
delay(200);
```

```
screen.text("Tampere University of", 15, 90);
screen.text("Technology (TUT)", 35, 100);
screen.rect(60, 123, 5, 5);
delay(200);
screen.rect(90, 123, 5, 5);
delay(200);
screen.rect(65, 118, 5, 5);
delay(200);
screen.rect(85, 118, 5, 5);
delay(200);
screen.rect(70, 123, 5, 5);
delay(200);
screen.rect(80, 123, 5, 5);
delay(200);
screen.rect(75, 118, 5, 5);
delay(1000);
screen.background(0,0,77);  // Refresh background
} // End void setup()

void loop() {
Serial.write(num);
if(digitalRead(A4)==HIGH){ // Mode selection
  delay(500);
  var++;
  if(var>4){var=0;}}
switch (var) {

  case 0: // INSTRUCTIONS_____________________________________________________
  if(c==0){
  num=255; // Refresh speed residual from automatic mode
  screen.background(0,0,77);
  screen.stroke(255,128,0);
  screen.text("<-SLOTPIRAL R-EVOLUTION->", 5, 3);
  screen.text("Press [M] to select mode:", 5, 20);
  screen.text("1. linear ", 10, 35);
  screen.text("2. Logarithmic ", 10, 50);
  screen.text("3. Kids ", 10, 65);
  screen.text("4. Automatic ", 10, 80);
  screen.fill(255,255,255); // Flag checkered
  for(int i=0; i<8; i++){screen.rect(10*i, 123, 5, 5);}
  for(int i=0; i<8; i++){screen.rect(5+10*i, 118, 5, 5);}
  for(int i=0; i<9; i++){screen.rect(10*i, 113, 5, 5);}
  for(int i=0; i<8; i++){screen.rect(5+10*i, 108, 5, 5);}
  for(int i=0; i<9; i++){screen.rect(10*i, 103, 5, 5);}
  screen.fill(0,0,0); // Road
  screen.circle(160, 124, 80);
  screen.fill(0,0,77);
  screen.circle(160, 124, 27);
  screen.fill(255,255,255); // Flag checkered
```

```
  for(int i=0; i<3; i++){screen.rect(135+10*i, 123, 5, 5);}
  for(int i=0; i<2; i++){screen.rect(140+10*i, 118, 5, 5);}
  for(int i=0; i<3; i++){screen.rect(135+10*i, 113, 5, 5);}
  for(int i=0; i<2; i++){screen.rect(140+10*i, 108, 5, 5);}
  for(int i=0; i<2; i++){screen.rect(145+10*i, 103, 5, 5);}
  screen.fill(255,0,0); // Red car
  screen.noStroke();
  screen.rect(105, 88, 20, 33);
  screen.fill(150,150,150);
  screen.rect(103, 94, 3, 5);
  screen.rect(103, 111, 3, 5);
  screen.rect(124, 94, 3, 5);
  screen.rect(124, 111, 3, 5);
  screen.stroke(255,255,255);
  screen.fill(255,0,0);
  screen.circle(115, 105, 6);
  screen.text("7", 113, 102);
  screen.stroke(177,0,0);
  screen.line(108, 96, 122, 96);
  screen.line(108, 114, 122, 114);
  screen.stroke(255,128,0); }
  c=1; // Graphical control variable disable
break;

  case 1: // MODE LINEAR_____________________________________________________
  Serial.write(num);
  if(a==0){screen.background(0,0,77); // Graphic screen
    screen.fill(64,64,64);
    screen.rect(0, 0, 160, 13);
    screen.stroke(100,100,100);
    screen.circle(25, 135, 27);
    screen.circle(80, 135, 30);
    screen.circle(135, 135, 27);
    screen.stroke(255,128,0);
    screen.text("<-SLOTPIRAL R-EVOLUTION->", 5, 3);
    screen.stroke(255,255,255);
    screen.line(15, 90, 145, 90); // Axis X
    screen.line(15, 89, 145, 89);
    screen.text("t", 152, 86);
    screen.line(14, 91, 14, 18); // Axis Y
    screen.line(15, 18, 15, 90);
    screen.text("v", 4, 18);
    screen.text("Mode: linear", 40, 93);
    a=1; // Graphical control variable disable
    b=0; // Graphical control variable enable
    screen.noStroke(); // Speedometer silhouette
    screen.fill(64,64,64);
    for(int i=0; i<13; i++){screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
  }
```

```
  while(digitalRead(A0)==HIGH){ // Acceleration
  do{
    if(digitalRead(A3)==HIGH){ // Pass
      num=171+stlin; // Pass constant + residual throttle
      screen.stroke(255,255,0);
      screen.text("PASS", 67, 116);
      screen.noStroke();
    for(int i=0; i<7; i++){
      screen.fill(255,255-17*i,0);
      screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
    for(int i=7; i<13; i++){
      screen.fill(64,64,64);
      screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
    }else{screen.stroke(102,102,0);
      screen.text("PASS", 67, 116);
      screen.noStroke();}
    if(digitalRead(A2)==HIGH){ // Turbo
      num=64;
      screen.stroke(255,0,0);
      screen.text("TURBO", 120, 116);
      screen.noStroke();
      for(int i=0; i<13; i++){
        screen.fill(255,255-17*i,0);
        screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
    }else{screen.stroke(102,0,0);
      screen.text("TURBO", 120, 116);
      screen.noStroke();}
    if(digitalRead(A1)==HIGH){ // Brake
      num=255+stlin; // Stop + residual throttle
      screen.stroke(0,255,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();
      screen.fill(64,64,64);
    for(int  i=0;  i<13;  i++){screen.rect(16+i*9,  85-5*i,  10,  4+5*i);}  //
Speedometer silhouette
    }else{screen.stroke(0,102,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();}

    num-=stlin; // Increase speed
    delay(100); // Regule speed
    if(num<64){num=64;} // Maximum speed fixed

    Serial.write(num); // Speedometer
    if(num<=255&&num>242){screen.fill(255,255,0); screen.rect(16, 85, 10, 4);}
    if(num<=242&&num>229){screen.fill(255,238,0); screen.rect(25, 80, 10, 9);}
    if(num<=229&&num>216){screen.fill(255,221,0); screen.rect(34, 75, 10, 14);}
    if(num<=216&&num>203){screen.fill(255,204,0); screen.rect(43, 70, 10, 19);}
    if(num<=203&&num>190){screen.fill(255,187,0); screen.rect(52, 65, 10, 24);}
```

```
  if(num<=190&&num>177){screen.fill(255,170,0); screen.rect(61, 60, 10, 29);}
  if(num<=177&&num>164){screen.fill(255,153,0); screen.rect(70, 55, 10, 34);}
  if(num<=164&&num>151){screen.fill(255,136,0); screen.rect(79, 50, 10, 39);}
  if(num<=151&&num>138){screen.fill(255,119,0); screen.rect(88, 45, 10, 44);}
  if(num<=138&&num>125){screen.fill(255,102,0); screen.rect(97, 40, 10, 49);}
  if(num<=125&&num>112){screen.fill(255,85,0); screen.rect(106, 35, 10, 54);}
  if(num<=112&&num>99){screen.fill(255,68,0); screen.rect(115, 30, 10, 59); }
  if(num<=99&&num>63){screen.fill(255,51,0); screen.rect(124, 25, 10, 64);}
}while(digitalRead(A0)==HIGH); // End do-while structure
} // End while structure

do{ // Deceleration
  if(digitalRead(A3)==HIGH){ // Pass
    num=171-sblin; // Pass constant - residual brake
    screen.stroke(255,255,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();
    for(int i=0; i<7; i++){
      screen.fill(255,255-17*i,0);
      screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
    for(int i=7; i<13; i++){
      screen.fill(64,64,64);
      screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
  }else{screen.stroke(102,102,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();}
  if(digitalRead(A2)==HIGH){ // Turbo
    num=64-sblin; // Maximum speed - residual brake
    screen.stroke(255,0,0);
    screen.text("TURBO", 120, 116);
    screen.noStroke();
    for(int i=0; i<13; i++){
      screen.fill(255,255-17*i,0);
      screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
  }else{screen.stroke(102,0,0);
    screen.text("TURBO", 120, 116);
    screen.noStroke();}
  if(digitalRead(A1)==HIGH){ // Brake
    num=255+stlin; // Stop + residual throttle
    screen.stroke(0,255,0);
    screen.text("BRAKE", 10, 116);
    screen.noStroke();
    screen.fill(64,64,64);
    for(int i=0; i<13; i++){screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
    }else{screen.stroke(0,102,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();}

  num+=sblin; // Decrease speed
  delay(100); // Regule brake
```

```
    if(num>255){num=255;} // Stop fixed

    Serial.write(num); // Speedometer
    if(num==255){screen.fill(64,64,64); screen.rect(16, 85, 10, 4);}
    if(num<=254&&num>242){screen.fill(64,64,64); screen.rect(25, 80, 10, 9);}
    if(num<=242&&num>229){screen.fill(64,64,64); screen.rect(34, 75, 10, 14);}
    if(num<=229&&num>216){screen.fill(64,64,64); screen.rect(43, 70, 10, 19);}
    if(num<=216&&num>203){screen.fill(64,64,64); screen.rect(52, 65, 10, 24);}
    if(num<=203&&num>190){screen.fill(64,64,64); screen.rect(61, 60, 10, 29);}
    if(num<=190&&num>177){screen.fill(64,64,64); screen.rect(70, 55, 10, 34);}
    if(num<=177&&num>164){screen.fill(64,64,64); screen.rect(79, 50, 10, 39);}
    if(num<=164&&num>151){screen.fill(64,64,64); screen.rect(88, 45, 10, 44);}
    if(num<=151&&num>138){screen.fill(64,64,64); screen.rect(97, 40, 10, 49);}
    if(num<=138&&num>125){screen.fill(64,64,64); screen.rect(106, 35, 10, 54);}
    if(num<=125&&num>112){screen.fill(64,64,64); screen.rect(115, 30, 10, 59);}
    if(num<=112&&num>90){screen.fill(64,64,64); screen.rect(124, 25, 10, 64);}
  }while(digitalRead(A0)==LOW&&num!=255); // End do-while structure
break;

  case 2: // MODE LOGARITHMIC_____________________________________________
  Serial.write(num);
  if(b==0){screen.stroke(0,0,77); // Refresh graphic screen
    screen.text("Mode: linear", 40, 93);
    screen.fill(0,0,77);
    for(int i=0; i<13; i++){screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
  }
  b=1; // Graphical control variable disable
  a=0; // Graphical control variable enable
  if(num==255){ screen.noStroke();
    screen.fill(64,64,64);
    for(int  i=1;  i<14;  i++){screen.rect(7+i*9,  82-int(51*log10(i)),  10,
7+int(51*log10(i)));}
    screen.stroke(255,255,255);
    screen.text("Mode: logarithmic", 30, 93);
  }

  while(digitalRead(A0)==HIGH){ // Acceleration
  do{
    if(digitalRead(A3)==HIGH){ // Pass
      n=3.7; // Pass logarithmic steps
      num=171+stlog; // Pass constant + residual throttle
      screen.stroke(255,255,0);
      screen.text("PASS", 67, 116);
      screen.noStroke();
      for(int i=1; i<8; i++){
        screen.fill(255,255-17*i,0);
        screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
      for(int i=7; i<14; i++){
        screen.fill(64,64,64);
        screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
```

```
    }else{screen.stroke(102,102,0);
      screen.text("PASS", 67, 116);
      screen.noStroke();}
    if(digitalRead(A2)==HIGH){ // Turbo
      num=64;
      screen.stroke(255,0,0);
      screen.text("TURBO", 120, 116);
      screen.noStroke();
    for(int i=1; i<14; i++){
      screen.fill(255,255-17*i,0);
      screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
    }else{screen.stroke(102,0,0);
      screen.text("TURBO", 120, 116);
      screen.noStroke();}
    if(digitalRead(A1)==HIGH){ // Brake
      n=1; // Initialize logarithmic steps
      num=255+stlog; // Stop + residual throttle
      screen.stroke(0,255,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();
      screen.fill(64,64,64);
      for(int  i=1;  i<14;  i++){screen.rect(7+i*9,  82-int(51*log10(i)),  10,
7+int(51*log10(i)));}
    }else{screen.stroke(0,102,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();}

    if(digitalRead(A1)==LOW){
      num=int(255-(cte*log10(n))); // Increase speed
      delay(80); // Regule brake
      n++;
      if(n>20){n=20;}} // Maximum limit speed

    Serial.write(num);
    if(num==255){screen.fill(255,255,0); screen.rect(16, 82, 10, 7);}
    if(num<=244&&num>200){screen.fill(255,238,0); screen.rect(25, 67, 10, 22);}
    if(num<=200&&num>180){screen.fill(255,221,0); screen.rect(34, 58, 10, 31);}
    if(num<=180&&num>160){screen.fill(255,204,0); screen.rect(43, 52, 10, 37);}
    if(num<=160&&num>150){screen.fill(255,187,0); screen.rect(52, 47, 10, 42);}
    if(num<=150&&num>135){screen.fill(255,170,0); screen.rect(61, 43, 10, 46);}
    if(num<=135&&num>120){screen.fill(255,153,0); screen.rect(70, 39, 10, 50);}
    if(num<=120&&num>110){screen.fill(255,136,0); screen.rect(79, 36, 10, 53);}
    if(num<=110&&num>100){screen.fill(255,119,0); screen.rect(88, 34, 10, 55);}
    if(num<=100&&num>90){screen.fill(255,102,0); screen.rect(97, 31, 10, 58);}
    if(num<=90&&num>80){screen.fill(255,85,0); screen.rect(106, 29, 10, 60);}
    if(num<=80&&num>70){screen.fill(255,68,0); screen.rect(115, 27, 10, 62);}
    if(num<=70&&num>63){screen.fill(255,51,0); screen.rect(124, 26, 10, 63);}
  } while(digitalRead(A0)==HIGH); // End do-while structure
  } // End while
```

```
  do{ // Deceleration
  if(digitalRead(A3)==HIGH){ // Pass
    n=4; // Pass logarithmic steps
    num=171-sblog; // Pass constant - residual brake
    screen.stroke(255,255,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();
    for(int i=1; i<8; i++){
      screen.fill(255,255-17*i,0);
      screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
    for(int i=7; i<14; i++){
      screen.fill(64,64,64);
      screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
  }else{screen.stroke(102,102,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();}
  if(digitalRead(A2)==HIGH){ // Turbo
    num=64-sblog; // Maximum speed - residual brake
    screen.stroke(255,0,0);
    screen.text("TURBO", 120, 116);
    screen.noStroke();
  for(int i=1; i<14; i++){
    screen.fill(255,255-17*i,0);
    screen.rect(7+i*9, 82-int(51*log10(i)), 10, 7+int(51*log10(i)));}
  }else{screen.stroke(102,0,0);
    screen.text("TURBO", 120, 116);
    screen.noStroke();}
  if(digitalRead(A1)==HIGH){ // Brake
    n=1; // Initialize logarithmic steps
    num=255+stlog; // Stop + residual throttle
    screen.stroke(0,255,0);
    screen.text("BRAKE", 10, 116);
    screen.noStroke();
    screen.fill(64,64,64);
    for(int  i=1;  i<14;  i++){screen.rect(7+i*9,  82-int(51*log10(i)),  10,
7+int(51*log10(i)));}
  }else{screen.stroke(0,102,0);
      screen.text("BRAKE", 10, 116);
      screen.noStroke();}

  if(n>1){n--;}
  num+=sblog; // Decrease speed
  delay(100); // Regule brake
  if(num>255){num=255;} // Stop fixed

  Serial.write(num); // Speedometer
  if(num==255){screen.fill(64,64,64); screen.rect(16, 82, 10, 7);}
  if(num<=254&&num>242){screen.fill(64,64,64); screen.rect(25, 67, 10, 22);}
  if(num<=242&&num>229){screen.fill(64,64,64); screen.rect(34, 58, 10, 31);}
  if(num<=229&&num>216){screen.fill(64,64,64); screen.rect(43, 52, 10, 37);}
```

```
  if(num<=216&&num>203){screen.fill(64,64,64); screen.rect(52, 47, 10, 42);}
  if(num<=203&&num>190){screen.fill(64,64,64); screen.rect(61, 42, 10, 47);}
  if(num<=190&&num>177){screen.fill(64,64,64); screen.rect(70, 39, 10, 50);}
  if(num<=177&&num>164){screen.fill(64,64,64); screen.rect(79, 35, 10, 54);}
  if(num<=164&&num>151){screen.fill(64,64,64); screen.rect(88, 33, 10, 56);}
  if(num<=151&&num>138){screen.fill(64,64,64); screen.rect(97, 31, 10, 58);}
  if(num<=138&&num>125){screen.fill(64,64,64); screen.rect(106, 28, 10, 61);}
  if(num<=125&&num>112){screen.fill(64,64,64); screen.rect(115, 27, 10, 62);}
  if(num<=112&&num>90){screen.fill(64,64,64); screen.rect(124, 25, 10, 64);}
  }while(digitalRead(A0)==LOW&&num!=255); // End do-while structure
break;

  case 3: // MODE KIDS_____________________________________________________
  Serial.write(num);
  if(a==0){screen.stroke(0,0,77); // Refresh graphic screen
    screen.text("Mode: logarithmic", 30, 93);
    screen.fill(0,0,77);
    for(int  i=1;  i<14;  i++){screen.rect(7+i*9,  81-int(51*log10(i)),  10,
8+int(51*log10(i)));}
    }
  a=1; // Graphical control variable disable
  b=0; // Graphical control variable enable
  if(num==255){screen.noStroke();
    screen.fill(64,64,64);
    for(int i=0; i<13; i++){screen.rect(16+i*9, 85-4*i, 10, 4+4*i);}
    screen.stroke(255,255,255);
    screen.text("Mode: kids", 50, 93);
  }

  while(digitalRead(A0)==HIGH){ // Acceleration
  do{
    if(digitalRead(A3)==HIGH){ // Pass
      num=171+stkid; // Pass constant + residual throttle
      screen.stroke(255,255,0);
      screen.text("PASS", 67, 116);
      screen.noStroke();
      for(int i=0; i<7; i++){
        screen.fill(255,255-17*i,0);
        screen.rect(16+i*9, 85-4*i, 10, 4+4*i);}
      for(int i=7; i<13; i++){
        screen.fill(64,64,64);
        screen.rect(16+i*9, 85-4*i, 10, 4+4*i);}
    }else{screen.stroke(102,102,0);
      screen.text("PASS", 67, 116);
      screen.noStroke();}
    if(digitalRead(A2)==HIGH){ // Turbo
      num=64; // Maximum speed
      screen.stroke(255,0,0);
      screen.text("TURBO", 120, 116);
      screen.noStroke();
```

```
    for(int i=0; i<13; i++){
      screen.fill(255,255-17*i,0);
      screen.rect(16+i*9, 85-4*i, 10, 4+4*i);}
  }else{screen.stroke(102,0,0);
    screen.text("TURBO", 120, 116);
    screen.noStroke();}
  if(digitalRead(A1)==HIGH){ // Brake
    num=255+stkid; // Stop + residual throttle
    screen.stroke(0,255,0);
    screen.text("BRAKE", 10, 116);
    screen.noStroke();
    screen.fill(64,64,64);
    for(int i=0; i<13; i++){screen.rect(16+i*9, 85-4*i, 10, 4+4*i);}
  }else{screen.stroke(0,102,0);
    screen.text("BRAKE", 10, 116);
    screen.noStroke();}

  num-=stkid; // Increase speed
  delay(100); // Regule throttle
  if(num<64){num=64;} // Maximum speed fixed

  Serial.write(num); // Speedometer
  if(num<=255&&num>242){screen.fill(255,255,0); screen.rect(16, 85, 10, 4);}
  if(num<=242&&num>229){screen.fill(255,238,0); screen.rect(25, 81, 10, 8);}
  if(num<=229&&num>216){screen.fill(255,221,0); screen.rect(34, 77, 10, 12);}
  if(num<=216&&num>203){screen.fill(255,204,0); screen.rect(43, 73, 10, 16);}
  if(num<=203&&num>190){screen.fill(255,187,0); screen.rect(52, 69, 10, 20);}
  if(num<=190&&num>177){screen.fill(255,170,0); screen.rect(61, 65, 10, 24);}
  if(num<=177&&num>164){screen.fill(255,153,0); screen.rect(70, 61, 10, 28);}
  if(num<=164&&num>151){screen.fill(255,136,0); screen.rect(79, 57, 10, 32);}
  if(num<=151&&num>138){screen.fill(255,119,0); screen.rect(88, 53, 10, 36);}
  if(num<=138&&num>125){screen.fill(255,102,0); screen.rect(97, 49, 10, 40);}
  if(num<=125&&num>112){screen.fill(255,85,0); screen.rect(106, 45, 10, 44);}
  if(num<=112&&num>99){screen.fill(255,68,0); screen.rect(115, 41, 10, 48); }
  if(num<=99&&num>63){screen.fill(255,51,0); screen.rect(124, 37, 10, 52);}
}while(digitalRead(A0)==HIGH); // End do-while structure
} // End while structure

do{ // Deceleration
if(digitalRead(A3)==HIGH){ // Pass
  num=171-sbkid; // Pass constant - residual brake
  screen.stroke(255,255,0);
  screen.text("PASS", 67, 116);
  screen.noStroke();
  for(int i=0; i<7; i++){
    screen.fill(255,255-17*i,0);
    screen.rect(16+i*9, 85-4*i, 10, 4+4*i);}
  for(int i=7; i<13; i++){
    screen.fill(64,64,64);
    screen.rect(16+i*9, 85-4*i, 10, 4+4*i);}
```

```
  }else{screen.stroke(102,102,0);
    screen.text("PASS", 67, 116);
    screen.noStroke();}
  if(digitalRead(A2)==HIGH){ // Turbo
   num=64-sbkid; // Maximum speed - residual brake
   screen.stroke(255,0,0);
   screen.text("TURBO", 120, 116);
   screen.noStroke();
  for(int i=0; i<13; i++){
     screen.fill(255,255-17*i,0);
     screen.rect(16+i*9, 85-4*i, 10, 4+4*i);}
  }else{screen.stroke(102,0,0);
    screen.text("TURBO", 120, 116);
    screen.noStroke();}
  if(digitalRead(A1)==HIGH){ // Brake
    num=255+stkid; // Stop + residual throttle
    screen.stroke(0,255,0);
    screen.text("BRAKE", 10, 116);
    screen.noStroke();
    screen.fill(64,64,64);
    for(int i=0; i<13; i++){screen.rect(16+i*9, 85-4*i, 10, 4+4*i);}
  }else{screen.stroke(0,102,0);
    screen.text("BRAKE", 10, 116);
    screen.noStroke();}

  num+=sbkid; // Decrease speed
  delay(100); // Regule brake
  if(num>255){num=255;} // Stop fixed

  Serial.write(num); // Speedometer
  if(num==255){screen.fill(64,64,64); screen.rect(16, 85, 10, 4);}
  if(num<=254&&num>242){screen.fill(64,64,64); screen.rect(25, 81, 10, 8);}
  if(num<=242&&num>229){screen.fill(64,64,64); screen.rect(34, 77, 10, 12);}
  if(num<=229&&num>216){screen.fill(64,64,64); screen.rect(43, 73, 10, 16);}
  if(num<=216&&num>203){screen.fill(64,64,64); screen.rect(52, 69, 10, 20);}
  if(num<=203&&num>190){screen.fill(64,64,64); screen.rect(61, 65, 10, 24);}
  if(num<=190&&num>177){screen.fill(64,64,64); screen.rect(70, 61, 10, 28);}
  if(num<=177&&num>164){screen.fill(64,64,64); screen.rect(79, 57, 10, 32);}
  if(num<=164&&num>151){screen.fill(64,64,64); screen.rect(88, 53, 10, 36);}
  if(num<=151&&num>138){screen.fill(64,64,64); screen.rect(97, 49, 10, 40);}
  if(num<=138&&num>125){screen.fill(64,64,64); screen.rect(106, 45, 10, 44);}
  if(num<=125&&num>112){screen.fill(64,64,64); screen.rect(115, 41, 10, 48);}
  if(num<=112&&num>90){screen.fill(64,64,64); screen.rect(124, 37, 10, 52);}
  } while(digitalRead(A0)==LOW&&num!=255); // End do-while structure
break;

  case 4: // MODE AUTOMATIC_____________________________________________________

  Serial.write(num);
  if(b==0){ screen.stroke(0,0,77); // Refresh graphic screen
```

```
  screen.text("Mode: kids", 50, 93);
  screen.fill(0,0,77);
  screen.stroke(64,64,64);
  screen.text("PASS", 67, 116);
  screen.text("BRAKE", 10, 116);
  screen.text("TURBO", 120, 116);
  for(int i=0; i<13; i++){screen.rect(16+i*9, 85-4*i, 10, 4+4*i);}
}
b=1; // Graphical control variable disable
c=0; // Graphical control variable enable
a=0; // Graphical control variable enable
if(num==255){screen.stroke(255,255,255);
screen.fill(64,64,64);
for(int i=0; i<13; i++){screen.rect(16+i*9, 85-5*i, 10, 4+5*i);}
screen.stroke(255,255,255);
screen.text("Mode: automatic", 35, 93);
}

int cc=random(0,10); // Random pass algorithm
if(cc<6){delay(200);
  screen.stroke(102,102,0);
  screen.text("AUTOPASS", 57, 116);
  screen.setTextSize(2);
  screen.text("<         >", 15, 112);
  screen.noStroke();
  screen.setTextSize(1);
}delay(200);
  screen.stroke(255,255,0);
  screen.text("AUTOPASS", 57, 116);
  screen.setTextSize(2);
  screen.text("<         >", 15, 112);
  screen.noStroke();
  screen.setTextSize(1);}

while(digitalRead(A0)==HIGH){ // Acceleration
  num-=staut;  // Increase speed
  delay(100); // Regule throttle
  if (num<64){num=64;} // Maximum speed fixed

  Serial.write(num); // Speedometer
  screen.stroke(255,255,255);
  if(num<=255&&num>242){screen.fill(255,255,0); screen.rect(16, 85, 10, 4);}
  if(num<=242&&num>229){screen.fill(255,238,0); screen.rect(25, 80, 10, 9);}
  if(num<=229&&num>216){screen.fill(255,221,0); screen.rect(34, 75, 10, 14);}
  if(num<=216&&num>203){screen.fill(255,204,0); screen.rect(43, 70, 10, 19);}
  if(num<=203&&num>190){screen.fill(255,187,0); screen.rect(52, 65, 10, 24);}
  if(num<=190&&num>177){screen.fill(255,170,0); screen.rect(61, 60, 10, 29);}
  if(num<=177&&num>164){screen.fill(255,153,0); screen.rect(70, 55, 10, 34);}
  if(num<=164&&num>151){screen.fill(255,136,0); screen.rect(79, 50, 10, 39);}
  if(num<=151&&num>138){screen.fill(255,119,0); screen.rect(88, 45, 10, 44);}
```

```
    if(num<=138&&num>125){screen.fill(255,102,0); screen.rect(97, 40, 10, 49);}
    if(num<=125&&num>112){screen.fill(255,85,0); screen.rect(106, 35, 10, 54);}
    if(num<=112&&num>99){screen.fill(255,68,0); screen.rect(115, 30, 10, 59); }
    if(num<=99&&num>63){screen.fill(255,51,0); screen.rect(124, 25, 10, 64);}
  } // End while structure


  while(digitalRead(A1)==HIGH&&num!=255){ // Deceleration
    num+=sbaut; // Decrease speed
    delay(100); // Regule throttle
    if(num>255){num=255;} // Stop fixed

    Serial.write(num); // Speedometer
    screen.stroke(255,255,255);
    if(num==255){screen.fill(64,64,64); screen.rect(16, 85, 10, 4);}
    if(num<=250&&num>241){screen.fill(64,64,64); screen.rect(25, 80, 10, 9);}
    if(num<=241&&num>228){screen.fill(64,64,64);  screen.rect(34, 75, 10, 14);}
    if(num<=228&&num>215){screen.fill(64,64,64); screen.rect(43, 70, 10, 19);}
    if(num<=215&&num>202){screen.fill(64,64,64); screen.rect(52, 65, 10, 24);}
    if(num<=202&&num>189){screen.fill(64,64,64); screen.rect(61, 60, 10, 29);}
    if(num<=189&&num>176){screen.fill(64,64,64); screen.rect(70, 55, 10, 34);}
    if(num<=176&&num>163){screen.fill(64,64,64); screen.rect(79, 50, 10, 39);}
    if(num<=163&&num>150){screen.fill(64,64,64); screen.rect(88, 45, 10, 44);}
    if(num<=150&&num>137){screen.fill(64,64,64); screen.rect(97, 40, 10, 49);}
    if(num<=137&&num>124){screen.fill(64,64,64); screen.rect(106, 35, 10, 54);}
    if(num<=124&&num>111){screen.fill(64,64,64); screen.rect(115, 30, 10, 59); }
    if(num<=111&&num>63){screen.fill(64,64,64); screen.rect(124, 25, 10, 64);}
  } // End while structure
break;
} // End switch structure
} // End void loop()
```

APPENDIX B: Source code of Slotpiral R-evolution receiver

```
//RECEIVER SLOTPIRAL R-EVOLUTION PROGRAM CODE
//GLOBAL VARIABLES
int num; // Counter variable
int k; // Decimal-binary conversion variable
int var; // Case structure variable

void setup() {
Serial.begin(9600); // Initialize serial communication
for(int i=2;i<=9;i++){pinMode(i, OUTPUT);} // Bits DAC D2-D9
pinMode(12, OUTPUT); // Pass high pulse
pinMode(10, OUTPUT); // Relay coil
pinMode(A0, INPUT); // Receiver mode push button (M)
} // End void setup()

void loop() {
if(digitalRead(A0)==HIGH){ // Mode selection
  delay(500);
  var+=1;
  if(var>1){var=0;}}
switch (var) {

  case 0: // ALL MODES_____________________________________________________
  if(Serial.available()>0){ // Receive data
    num=Serial.read(); // Receive counter variable
    Serial.print(num); // Show counter variable
    if(num!=255){digitalWrite(10,HIGH); // Tripping relay coil
    }else{digitalWrite(10, LOW);}
    if(num==171){digitalWrite(12,HIGH); // Pass enable
    }else{digitalWrite(12, LOW);} // Pass disable
  k=num; // Decimal-binary converter
  for(int i=2;i<=9;i++){
    digitalWrite(i, k%2);
    k=k/2;}
  } // End if structure
break;

  case 1: // AUTOMATIC MODE ONLY_____________________________________________
    if (Serial.available()>0){ // Receive data
    num=Serial.read(); // Receive counter variable
    Serial.print(num); // Show counter variable
    if(num==255){digitalWrite(10, LOW);
      digitalWrite(12,LOW);}
    if(num!=255){digitalWrite(10,HIGH); // Tripping relay coil
      int cc=random(0,5);
      if(cc<3){digitalWrite(12,LOW);
      delay(200);
    }else{digitalWrite(12,HIGH);
      delay(200);}}
    k=num; // Decimal-binary converter
    for(int i=2;i<=9;i++){
      digitalWrite(i, k%2);
      k=k/2;}
  } // End if
break;
} // End switch structure
} // End void loop()
```

APPENDIX C: Electronic schematics

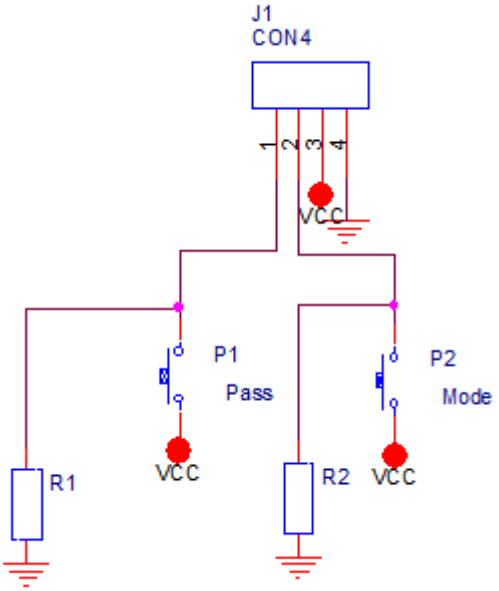

Figure 87. *Button L shield schematic*[2].

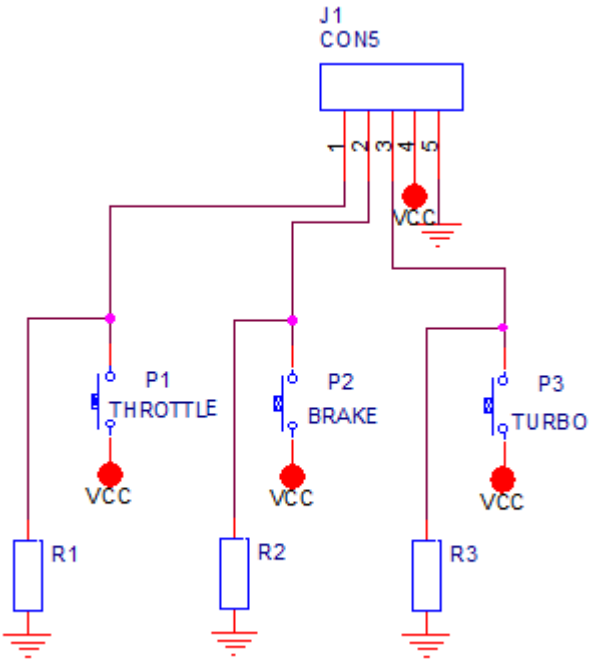

Figure 88. *Button R shield schematic*[2].

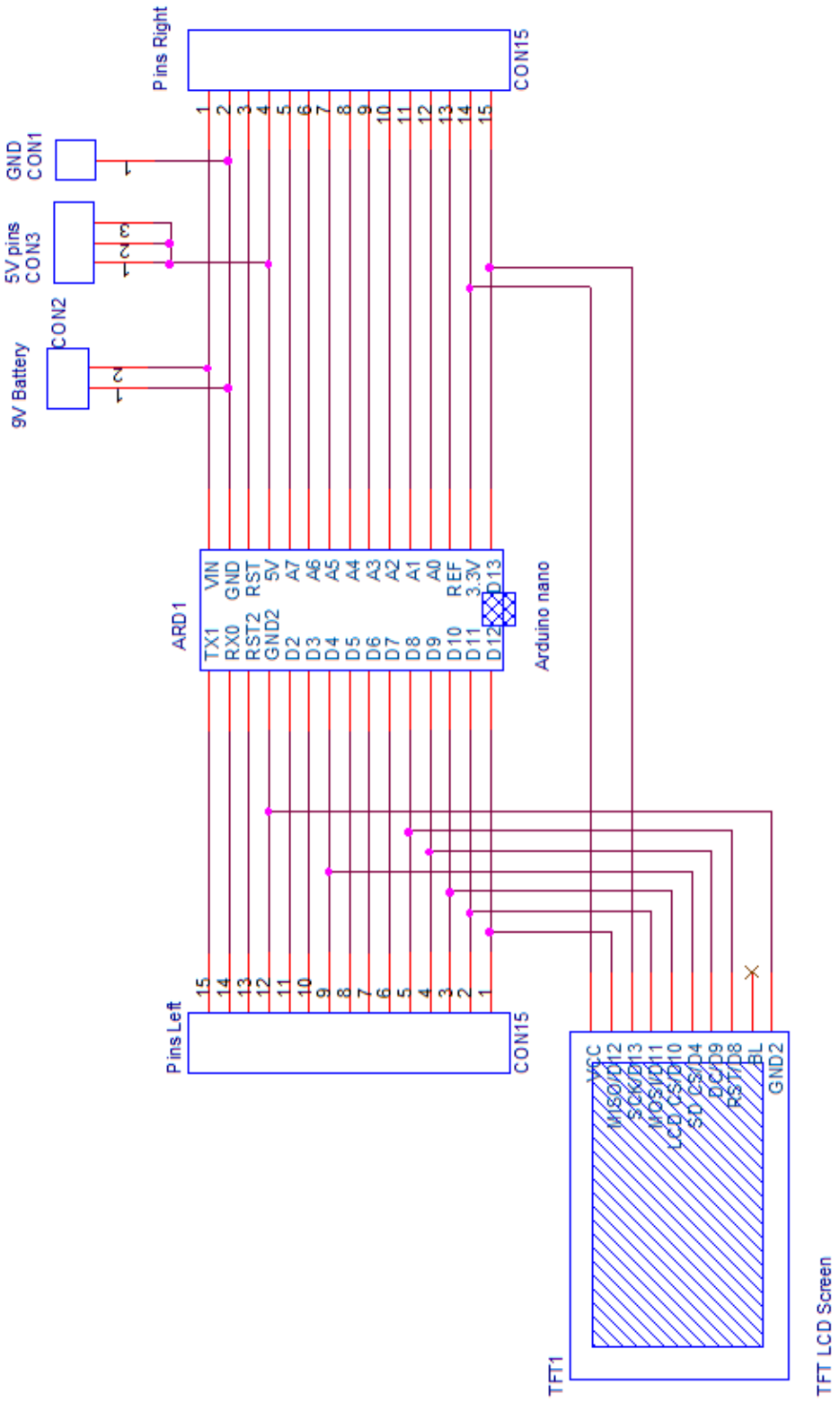

Figure 89. *Central shield schematic*[2].

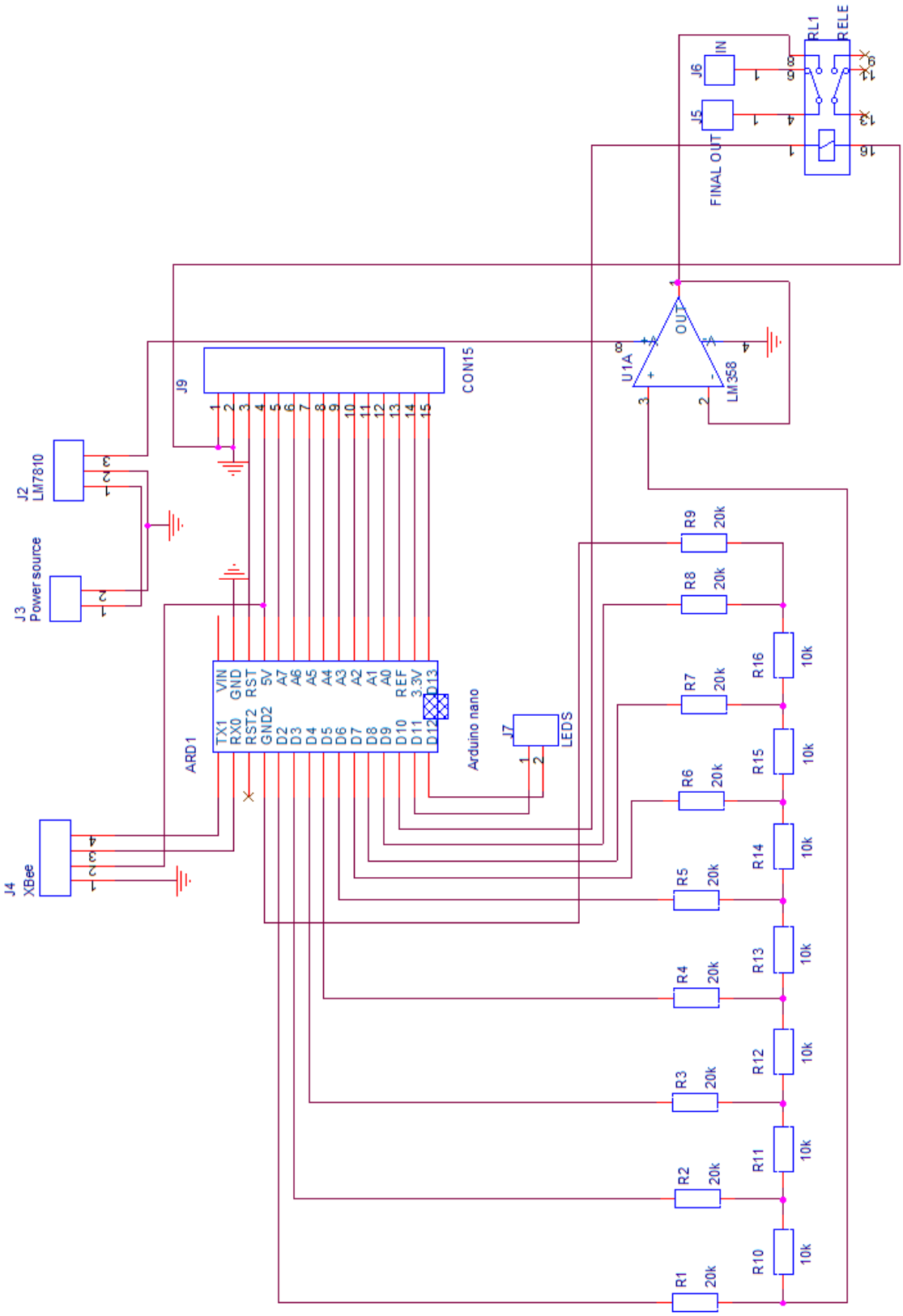

Figure 90. *DAC shield schematic*[2].

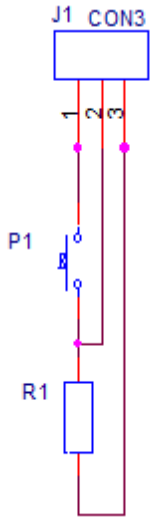

Figure 91. *PB shield schematic*[2].

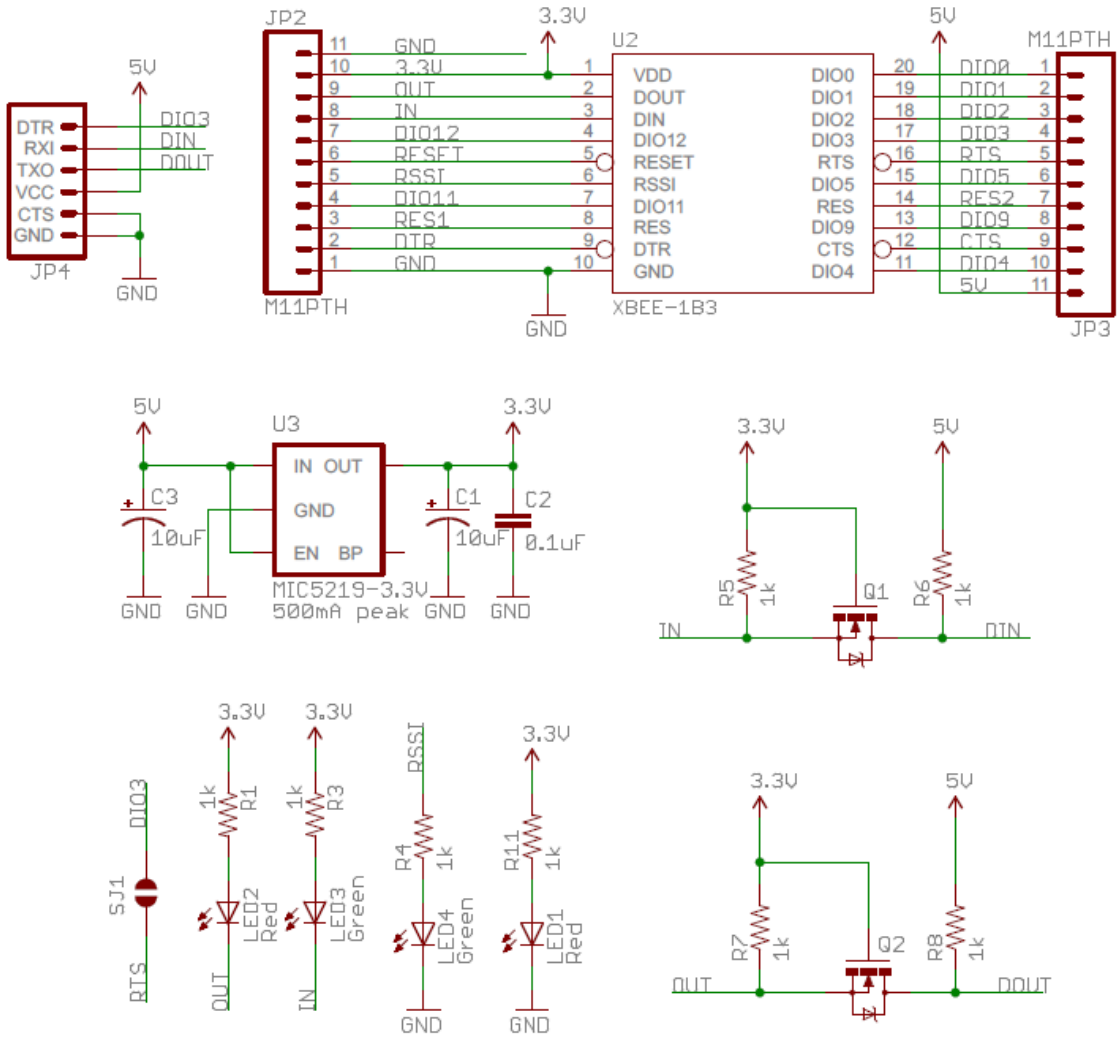

Figure 92. *XBee shield schematic* [51].